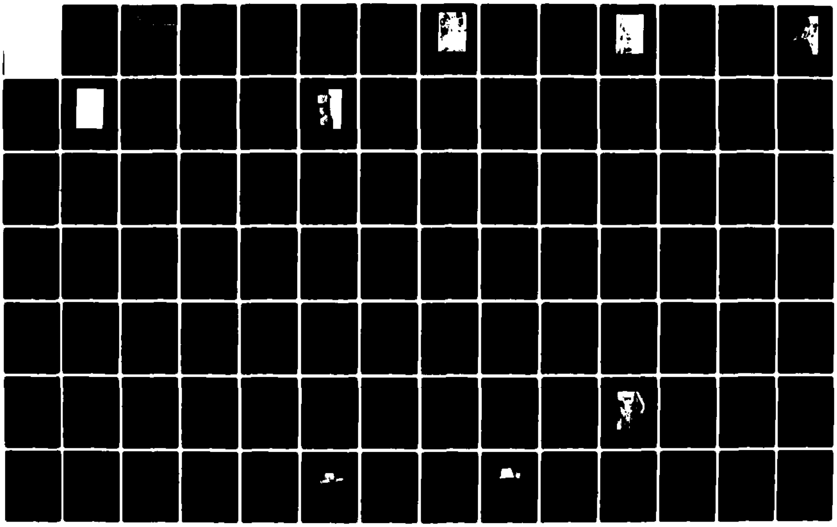
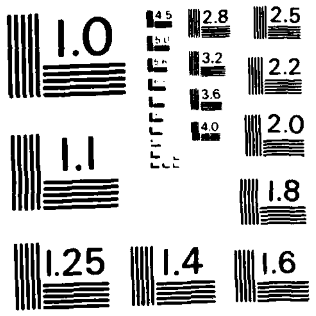
AD-A131 449 AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION(U) OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS R B MCGHEE ET AL. JUL 83 1/4

UNCLASSIFIED MDA903-82-K-0058 F/G 6/4 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5

RF Project 762945/714250
Semi-Annual Report

the
ohio
state
university

research foundation

1314 kinnear road
columbus, ohio
43212

AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION

Robert B. McGhee and Kenneth J. Waldron
Department of Mechanical Engineering

For the Period
October 1, 1982 - March 31, 1983

DEFENSE SUPPLY SERVICE
Washington, D.C. 20310

Contract No. MDA903-82-K-0058

July, 1983

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO. AD A131449

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION

5. TYPE OF REPORT & PERIOD COVERED
Semi-Annual Technical Report
Oct. 1, 1982 - Mar. 31, 1983

6. PERFORMING ORG. REPORT NUMBER
762943/714250

7. AUTHOR(s)
Robert B. McGhee and Kenneth J. Waldron

8. CONTRACT OR GRANT NUMBER(s)
Contract No.
MDA903-82-K-0058

9. PERFORMING ORGANIZATION NAME AND ADDRESS
The Ohio State University Research Foundation
1314 Kinnear Road
Columbus, Ohio 43212

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
DEPARTMENT OF THE ARMY, Defense Supply Service
Room 1D-245, The Pentagon
Washington, D.C. 20310

12. REPORT DATE
July, 1983

13. NUMBER OF PAGES
329

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Walking machine
Robotics
Off-road vehicles
Legged locomotion

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report summarizes research accomplished during the third six-month period of this three-year project. The research is aimed at field testing of an experimental vehicle making use of actively-controlled suspension units. The current design concept envisages locomotion without the use of wheels or tracks, but rather employs six active suspension elements, each providing support and propulsion analogously to a biological leg. This report describes

(continued)

Block 20 - Abstract (Continued)

a functioning breadboard leg and the associated vehicle frame. A scale model photograph is included to illustrate one candidate vehicle configuration.

Electrical system research has produced a breadboard computing system containing thirteen single-board computers (Intel 86/30). Breadboard software has been validated in simulation. Fusion of optically-sensed range data with other types of sensory information to provide automatic vehicle guidance in rough-terrain locomotion has been demonstrated experimentally in a laboratory-scale hexapod walking machine. An extensive appendix provides full details regarding this result.

Semi-Annual Technical Report

for

DARPA Contract MDA903-82-K-0058

AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION

prepared by

Robert B. McGhee and Kenneth J. Waldron

covering the period

October 1, 1982, through March 31, 1983

College of Engineering
The Ohio State University
Columbus, Ohio 43210

A

1 INTRODUCTION

Research under this contract began as scheduled on October 1, 1981. During the first eighteen months, no significant deviations occurred from the plan contained in our proposal dated May 13, 1981. Current project milestones are summarized in a list dated October 19, 1982, and attached to our Quarterly R & D Status Report for the period October 1, 1982 through December 31, 1982. All milestones on this list pertaining to the period covered by the present report have been met on schedule. The following pages present some details concerning work relating to each of these milestones. Further information is provided in the attached appendices and in theses, dissertations, published papers, and videotapes previously furnished to DARPA.

2. MECHANICAL SYSTEM RESEARCH

2.1 Breadboard Leg Tests

The breadboard leg system was extensively tested using a valve-controlled hydraulic actuation system. A bypass control valve was used on the drive actuator with conventional, four-way throttling valves on the leg lift and abduction-adduction actuators. The drive actuator was a balanced area actuator, while the other actuators were single-sided. All were linear actuators. They were supplied by variable displacement pumps operating in a pressure regulated mode. Figure 1 shows the breadboard leg mounted in its test stand.

The system was successfully operated using both analog and digital controllers. The latter was used during data collection runs. The digital stride cycle control program was successfully modified to minimize shock associated with changes in leg direction and to optimize dynamic response. The system was tested at equivalent ground speeds in excess of 3 mph. The leg was also static loaded in excess of its design load of 2000 lb. Details of

Figure 1: Breadboard leg mounted on test stand.

the test program and results may be found in the thesis "Modeling and Control of the Hydraulic Drive System for a Walking Machine Leg" by John F. Gardner.

A series of energy consumption tests were also run on the leg. The leg exhibited a specific resistance of 0.078. This figure includes actuator losses, but not pump or drive-train losses. Allowing for these, and for the overhead power needed to run the system electronics, a specific resistance comparable to that of tracked vehicles appears readily attainable.

Static and dynamic deflection studies have also been performed on the leg structure and compared to the results of finite element models. Structural stiffness is adequate.

2.2 Hydraulic Circuit Design Studies

Extensive design studies were performed on an alternative power transmission and actuation system concept using hydrostatic transmissions. Comparative cost, weight, and performance studies were conducted for the hydrostatic system and the best valve-controlled configuration. The hydrostatic system has lower cost, lighter weight, and less power demand. Its dynamic response is not markedly inferior, and it has the advantage of much greater simplicity and modularity in both hardware and in control. Therefore, it was decided to proceed to detailed design of a fully hydrostatic system. It is also planned to reconfigure the actuation system of the breadboard leg as a hydrostatic system for a further series of tests. Dynamic simulation and bench-test studies of series-bypass valve-controlled drive circuits are continuing. It is intended to augment them with dynamic simulation studies of the hydrostatic configuration.

2.3 Frame Tests

The vehicle frame was constructed. It weighs 315 lb. It was suspended from soft springs to approximate a free-free condition and dynamic vibration mode tests were performed using shaker excitation. The results showed reasonable agreement with those of finite element models used in the design. The frame is scheduled for static deflection tests during the next reporting period. Figure 2 shows the completed frame.

2.4 Leg Design

The design of the legs to be used on the vehicle is well advanced. Special actuators for the drive and leg lift have been designed. Paired actuators mounted on either side of the leg box will be used in both cases to prevent twisting moments on the roller assemblies. In order to save space, the equal area actuators will be mounted in both cases with the rods anchored to the leg box and the bodies of the cylinders coupled to the slides. Fluid will be fed through passages in the piston rods. This arrangement eliminates the need for flexible hydraulic connections and minimizes the active fluid volume.

It was decided to use a hinged ankle rather than the spherical joint used on the breadboard leg and to add a mechanism to maintain the foot in an attitude parallel to the vehicle body at all times. A roller chain mechanism has been designed for this purpose.

The new leg structure is fairly similar to that of the breadboard leg. The geometric proportions of the upper and lower links have have altered in accordance with the improved geometry detailed in Figure 3. This results in a larger, more symmetric working envelope. Figure 4 shows the revised leg mounting position and overall dimensions.

2.5 Other Work

A model of the current vehicle configuration concept was completed and is shown in Figure 5. It is one-tenth scale in size. The legs are movable

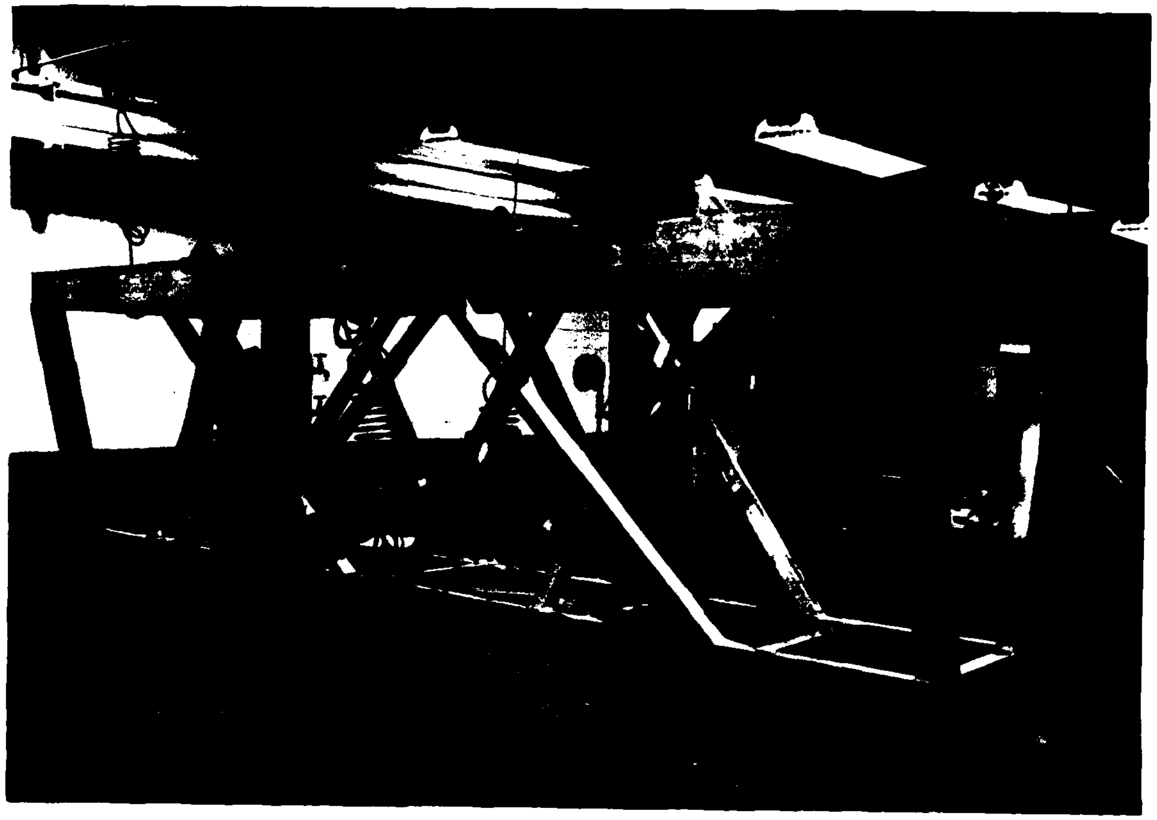


Figure 2: Completed frame mounted for free vibration mode tests.

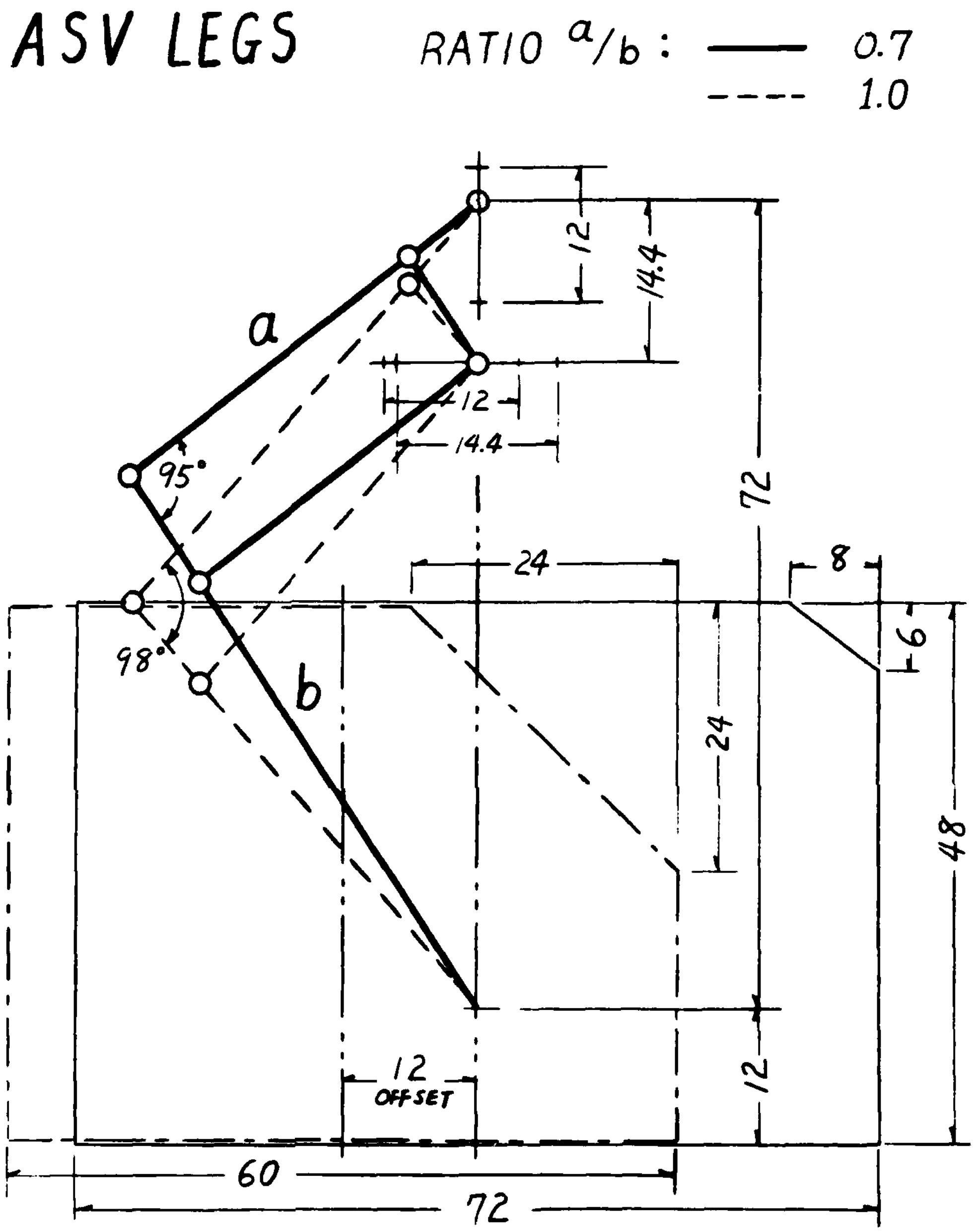


Figure 3: Comparison of new leg geometry (solid lines) with former leg geometry (dotted lines). The new working envelope is indicated by lightweight solid lines and the old by dashed lines.

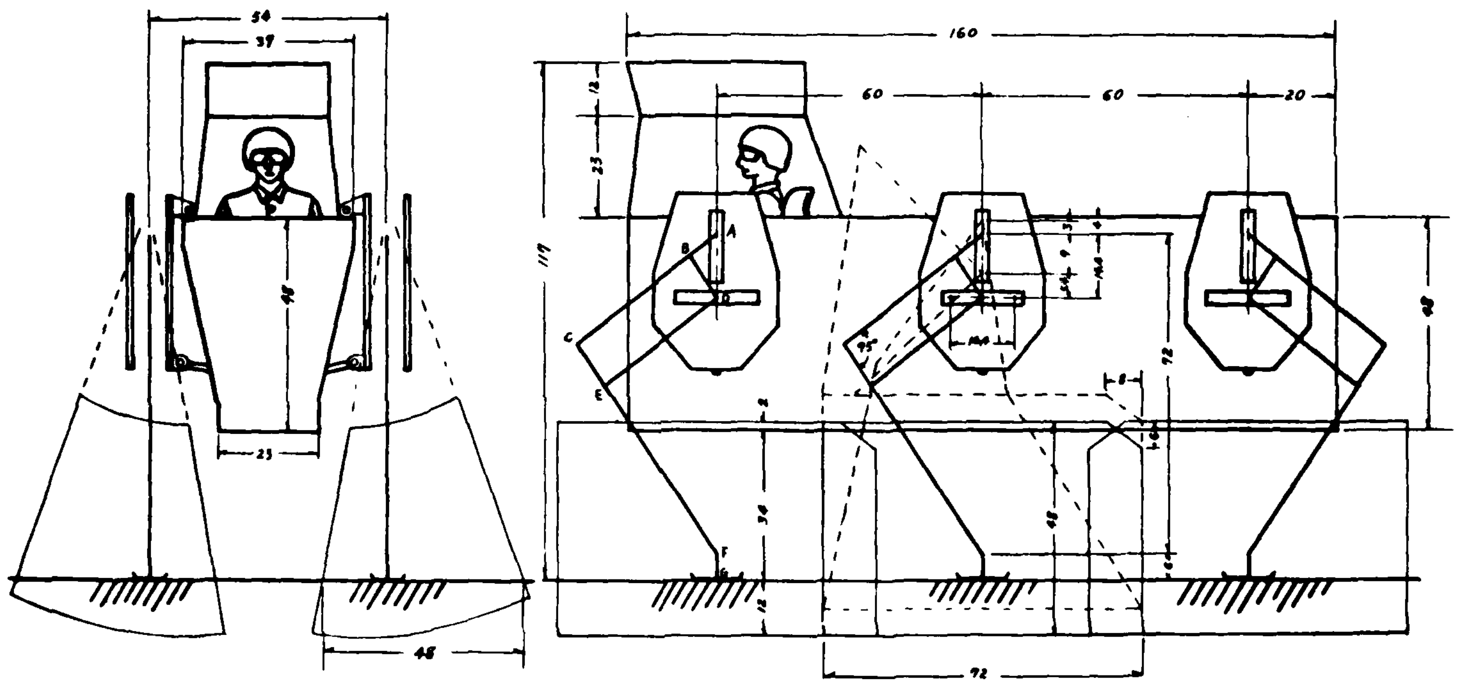


Figure 4: Dimensions and leg mounting positions for revised vehicle geometry.

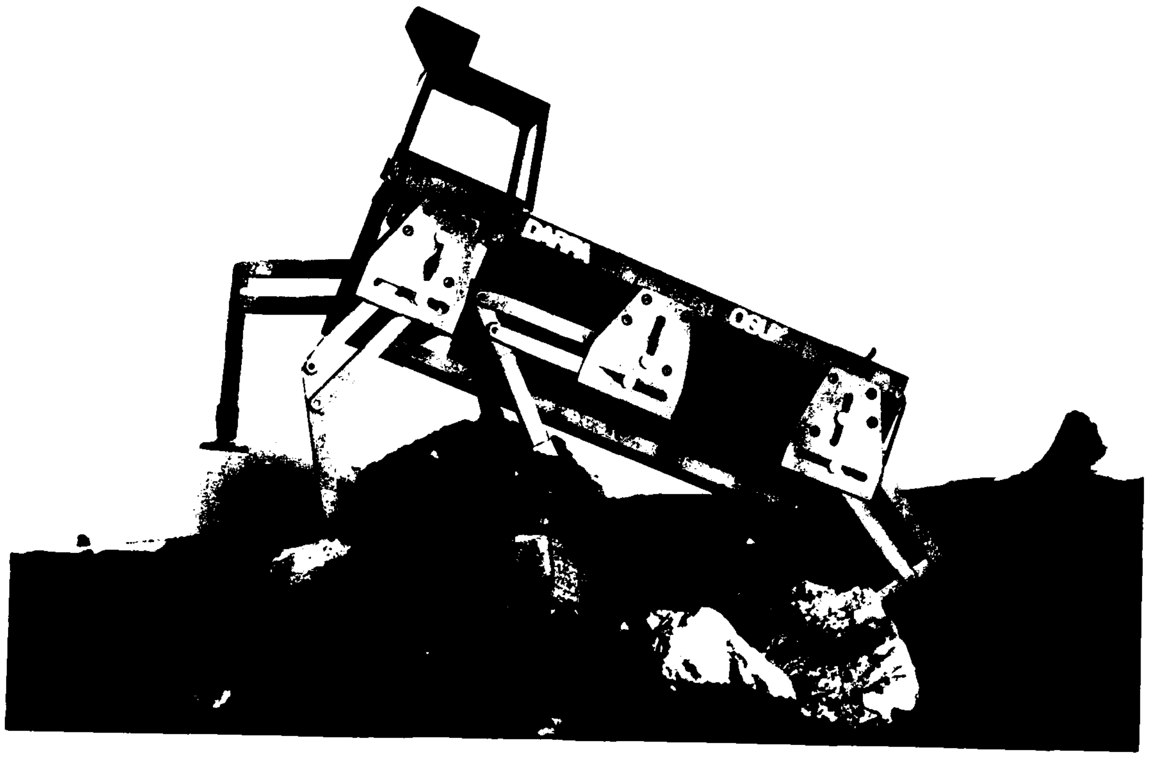


Figure 5: Model of ASV-84 on terrain board.

in all three degrees of freedom with friction drag to hold position. Table 1 lists the design goals for this vehicle concept.

Simulation studies of extreme obstacle crossing maneuvers were completed using the Evans and Sutherland PS 300 display system. Leg movements were determined by the programmer using trial and error. Thus, this can be regarded as a simulation of the precision footing mode. Crossings of a five foot, vertical-walled ditch and of a five foot vertical step in both directions were simulated. A frame from the display is shown in Figure 6.

3. ELECTRICAL SYSTEM RESEARCH

3.1 Breadboard Computer

Construction of a breadboard version of the ASV-84 control computer has been completed. This computer consists of nine Intel 86/30 single-board computer systems. Each board contains an 8086 processor, an 8087 co-processor, 16 channels of A to D conversion, and 8 channels of D to A conversion. Six of the computers are assigned to leg control; i.e., one for each leg. One computer is used for implementation of cockpit functions, and another is used as a coordination computer. Finally the remaining computer implements safety and diagnostics functions in addition to serving as a co-processor for the coordination computer.

The current hardware design for the Battelle guidance system makes use of four additional 86/30 boards, raising the total for the ASV-84 system to thirteen. The coordination computer is connected to the guidance computers by means of a parallel line unit and to all other computers by an Intel multibus. All multibus communications are initiated by the coordination computer, thereby eliminating the possibility of bus access conflicts.

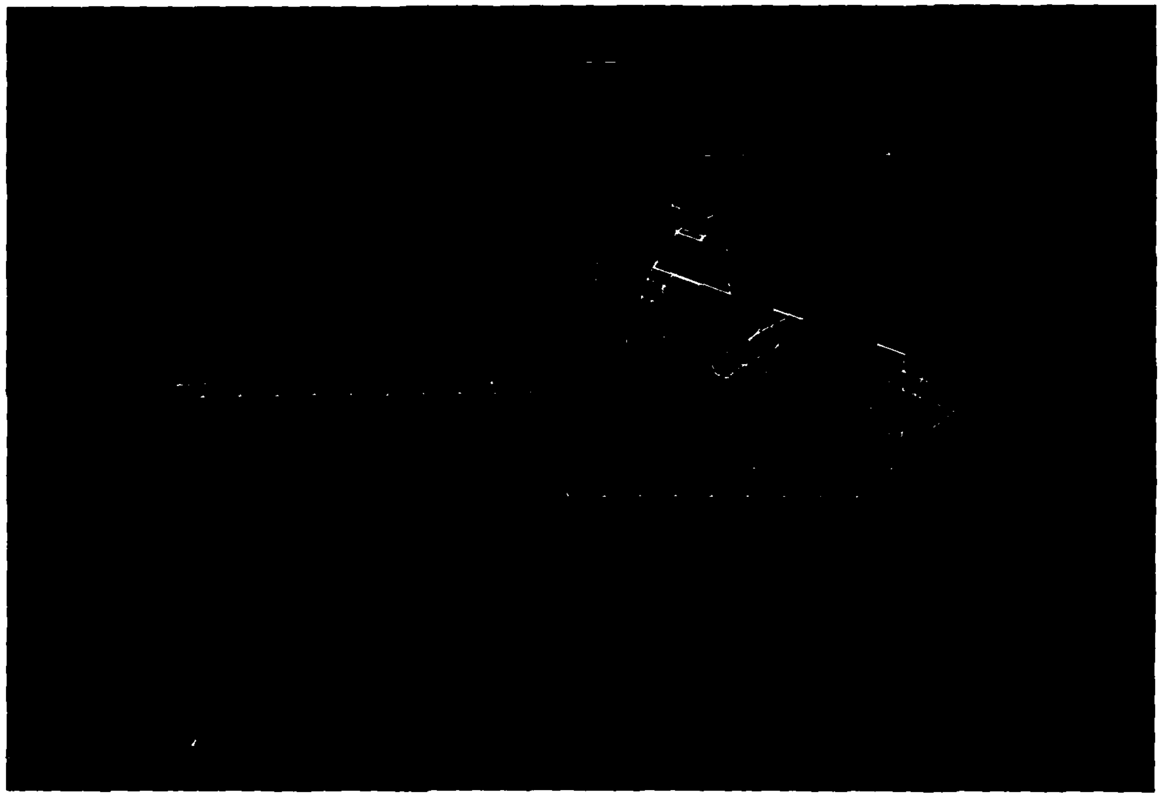


Figure 6: Simulation of the ASV-84 climbing a 5 ft vertical step. The diagram above the simulated vehicle is the associated support polygon.

Table 1

Design Goals for 1984 Proof-of-Concept Vehicle (ASV-84)

1. Level walking cruise speed of 5 mph.
2. Range of 50 miles without refueling.
3. Payload of 500 lb.
4. Vehicle dry-weight around 4000 lb.
5. Capable of climbing 60 percent slope on suitable soil.
6. Length about 14 ft., nominal height to top of frame around 7 ft., 10 ft. to top of terrain scanner.
7. Cross 7 ft. ditch with vertical walls, climb 5 ft. vertical step, traverse 4 ft. isolated vertical object.
8. Belly skids for parking and safety.

Portions of the breadboard computer have been used to control the electric monopod developed in the first year of this project. Anticipated cycle times were achieved, increasing our confidence in the suitability of the 86/30 board for hydraulic leg control. Details of this experiment can be found in the thesis of C.E. Barrientos, listed in Appendix 1.

The next stage of testing, presently scheduled for completion in September, 1983, will demonstrate that the breadboard computer possesses sufficient computing power to control the ASV-84 in real-time in the terrain following mode. These experiments will utilize the PDP-11/70 computer to simulate the vehicle, scanner, and terrain.

3.2 Binocular Vision System

The OSU Hexapod vehicle has been fitted with two CID television cameras to permit laboratory experiments relating to the integration of optically-sensed range information with ground reaction force and body attitude measurements. The two cameras derive range data by triangulation on candidate footholds designated by a human operator using a hand-held He-Ne laser. The system software then accepts or rejects such footholds based on the kinematic capabilities of the machine. Accepted footholds are used by succeeding legs by means of a "follow-the-leader" gait. Details are provided in the attached Appendix 3.

The primary significance of this experiment is that it demonstrates in a physical experiment the feasibility of the methods we propose for "fusing" the output of the Battelle guidance computer together with other sensory information to smoothly guide the ASV-84 machine over rough terrain. A videotape documenting this capability was provided to DARPA in December, 1982. A secondary importance of the results obtained is that follow-the-leader gaits seem to be well suited to climbing over large obstacles in a "precision footing" mode of control.

3.3 Safety and Diagnostics Software

A set of definitions and general characteristics of the ASV-84 safety and diagnostics software has been developed. This information was presented at a design review held in Naples, Florida on March 8-9, and subsequently adopted as a guide for the development of more detailed software specifications and vehicle operating procedures. This material is included in this report as Appendix 2.

3.4 Terrain-Following Software Test

The compatibility of Battelle guidance software and OSU autopilot software for the ASV-84 has been verified by a simulation test on the PDP-11/70 computer. This test included climbing slopes and stepping over ditches automatically. The response of the vehicle was presented in real-time on a vector graphics unit. For more detailed understanding, any selected frame of the vector graphics display can be converted to shaded color graphics. Figure 7 shows one frame from the latter type of display. Work in progress will permit the extension of this capability to the production of motion pictures of shaded color graphics representations of vehicle motion.

The simulation tests revealed only minor incompatibilities between OSU and Battelle software and these were all corrected. The next stage of testing, presently scheduled for September, 1983, will include demonstration of hardware compatibility between the OSU and Battelle components of the breadboard computer.

3.5 Proximity Sensors

An ultrasonic proximity sensing system suitable for laboratory use with the OSU Hexapod has been designed and tested. Results are fully satisfactory so long as the incidence angle between the terrain and sensor is restricted to less than abou 30° from vertical. When this angle is exceeded, insufficient

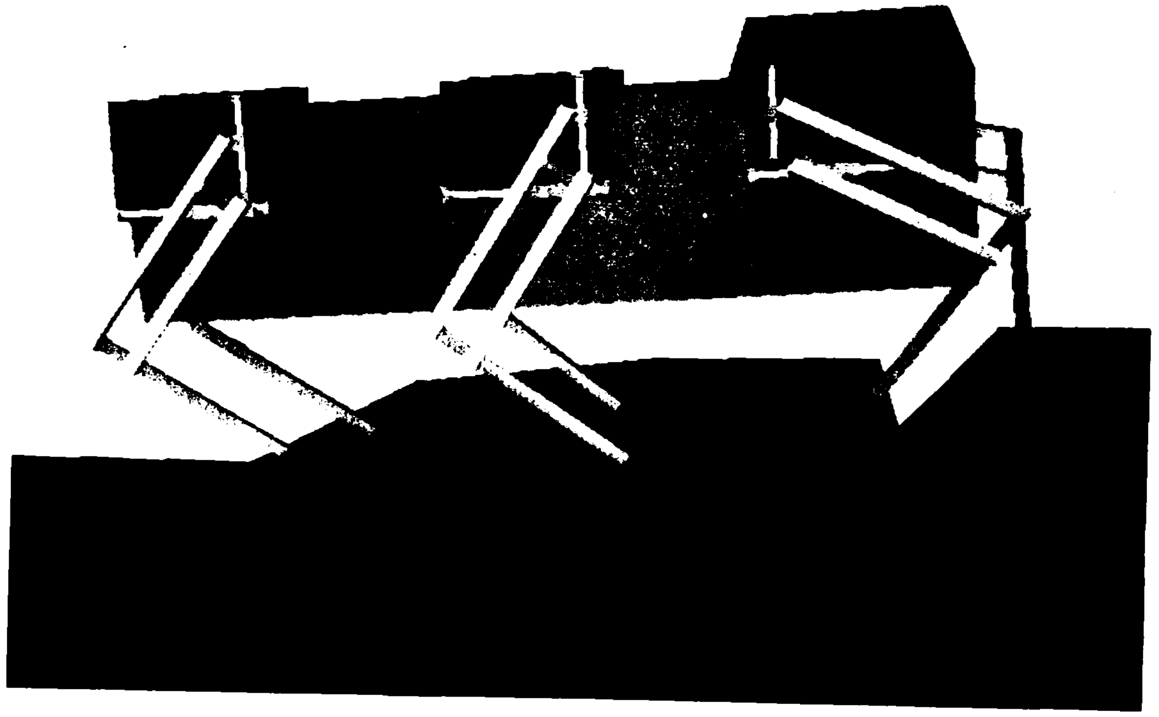


Figure 7: Single frame from shaded color graphic display of ASV-84 guidance simulation.

energy is reflected back to the receiver, and distance information is lost.

It is felt that rough-terrain locomotion in the close maneuvering mode can be accomplished in the laboratory using the present sensor design. This will be accomplished by choosing leg trajectories such that incidence angle constraints are satisfied. Additional sensing systems are now being constructed and installed so that each leg of the OSU Hexapod will be able to measure its elevation above the terrain. It is expected that locomotion using this type of feedback will be demonstrated by September, 1983.

With respect to the ASV-84, we anticipate that a more advanced type of proximity sensor will be needed. We intend to carry out an evaluation of the Kodak multiple frequency "chirp" sensors in the next six months using the breadboard leg as a test fixture. These tests will be conducted in a soil bin using various types of terrain to more realistically simulate an outdoor environment. We will also study the use of sensor arrays, both ultrasonic and optical, as a means of solving the incidence angle problem.

4. SUMMARY AND CONCLUSIONS

The research supported by this contract is on schedule and within budget. Breadboard leg tests have validated the pantograph leg geometry as being appropriate to full-scale vehicle operation. While valve-controlled operation produced adequate speed and precision of motion, issues of system weight, energy consumption, and ease of control now favor a hydrostatic, pump-controlled, hydraulic actuation system.

A baseline software system for the ASV-84 coordination computer has been successfully interfaced and tested with Battelle guidance software. The breadboard computer has been completed. Fusion of optically sensed range information with other sensory data for vehicle control has been demonstrated using the OSU Hexapod as a test bed. Providing subsystems from other contractors are delivered on time, there is at present no reason to doubt that a successful outdoor demonstration of terrain-following locomotion will be achieved as planned during August, 1984.

Appendix 1

List of Publications

3/31/83

RESEARCH PUBLICATIONS SUPPORTED BY CONTRACT MDA903-82-K-0058

College of Engineering
The Ohio State University
Columbus, Ohio 43210

The following theses, dissertations, and papers have been produced with the support of DARPA Contract MDA903-82-K-0058 since its inception on October 1, 1981. Copies of most of these documents are available upon request.

1. Waldron, K.J., Frank, A.A., and Srinivasan, K., "The Use of Mechanical Energy Storage in an Unconventional, Rough-Terrain Vehicle," 17th Intersociety Energy Conversion Engineering Conference, Los Angeles, California, August 8-13, 1982.

2. Brown, F.T., Dynamic Study of a Four-Bar Linkage Walking Machine Leg, M.S. thesis, The Ohio State University, August, 1982.

3. Ju, J.T., Safety Checking System with Voice Response for the OSU Hexapod, M.S. thesis, The Ohio State University, August, 1982.

4. Pugh, D.R., An Autopilot for a Terrain-Adaptive Hexapod Vehicle, M.S. thesis, The Ohio State University, August, 1982.

5. Tsai, C.K., Computer Control Design of an Energy-Efficient Leg, M.S. thesis, The Ohio State University, August, 1982.

6. Srinivasan, K., Waldron, K.J., and Dworak, J.A., "The Design and Evaluation of a Hydraulic Actuation System for a Legged Rough-Terrain Vehicle," presented at ASME Winter Annual Meeting, Phoenix, November 14-16, 1982, Robotics Research and Advanced Applications, ASME, 1982.

7. Waldron, K.J., Song, S.M., Vohnout, V.J., and Kinzel, G.L., "Computer-Aided Design of a Leg of an Energy-Efficient Walking Machine," Proceedings of 7th Applied Mechanisms Conference, Kansas City, December 7-9, 1981, pp. VIII-1 to VIII-8.

8. Kao, M.L., A Reliable Multi-Microcomputer System for Real Time Control, M.S. thesis, The Ohio State University, December, 1982.

9. Chang, T.W., Motion Planning for Locomotion of a Multilegged Robot over Uneven Terrain, M.S. thesis, The Ohio State University, December, 1982.

10. Cheng, T.T., Computer Simulation of the Dynamics and Control of an Energy-Efficient Robot Leg, M.S. thesis, The Ohio State University, December, 1982

11. Chang, T.S., Kinematic Simulation of an Adaptive Suspension Vehicle, M.S. thesis, The Ohio State University, December, 1982.

12. Barrientos, C.E., Development of a Multiple Microprocessor-Based Control for Legged Locomotion Processing Using Interactive Design tools, M.S. thesis, The Ohio State University, December, 1982.

13. Chuang, J.Y., Simulation of Load-Time History of an Adaptive Walking Vehicle by the Use of Rate Regulation, M.S. thesis, December, 1982.

14. Tsai, S.J., An Experimental Study of a Binocular Vision System for Rough Terrain Locomotion of a Hexapod Walking Robot, Ph.D. dissertation, May, 1983.

15. Gardner, J.F., Modeling and Control of the Hydraulic Drive System for a Walking Machine Leg, M.S. thesis, The Ohio State University, June, 1983.

16. McGhee, R.B., "Vehicular Legged Locomotion," Advances in Automation and Robotics, ed. by G.N. Saridis, Jai Press, Inc., 1983.

17. Vohnout, V.J., Mechanical Design of an Energy Efficient Robotic Leg for Use on a Multi-Legged Walking Vehicle, M.S. Thesis, The Ohio State University, June, 1982.

Appendix 2

Safety and Diagnostics Software Characteristics

SAFETY AND DIAGNOSTICS SYSTEM DEFINITIONS

SAFETY: Failure detection during vehicle operation followed by transition to a safe shut-down.

DIAGNOSTICS: Isolation of a failure to a replaceable module.

- Safety functions apply only when hydraulic power is applied to legs; i.e., safety software is real-time and on-line,

- Diagnostic procedures are off-line and interactive, but automated to the greatest extent possible. Generally applied only with vehicle in a safe halt state.

SHUT-DOWN MANEUVERS

CONTROLLED STOP: Vehicle comes to a gradual halt under computer control following detection of a non-catastrophic failure; e.g., loss of prime mover power (halt accomplished with energy stored in flywheel).

MOTION FREEZE: All actuators locked by closing servo valves following suspected leg control failure. Might be achieved by interrupting primary power to all valve drivers.

HARD LANDING: Drops machine onto belly skids in minimum time following malfunction of cockpit or guidance computer, vertical reference failure, incipient instability, operator panic, etc. Probably requires functioning coordination computer and leg servos.

FAILURE CATEGORIES

- SOFTWARE: Algorithms cannot be completely evaluated prior to field testing. Unexpected loads, operator error, new terrain conditions, etc., can lead to unsafe motion planning or motion execution.

- HARDWARE: Mechanical component failures can generally be detected by cross-checking of transducers and by out of range error signals. Not all computer failures can be directly detected.

- PRIMARY SYSTEMS: Safety computer can be directly connected via A to D converters to primary system transducers such as engine oil pressure, D.C. supply voltage, hydraulic pressure, vertical gyro, etc.

GENERAL APPROACH TO FAILURE DETECTION

- Safety function is distributed to minimize bus traffic.
- Detected errors set status bits in global memory of affected computer. No interrupts to higher level.
- Each computer checks its own peripherals. (A peripheral can be a transducer, a leg, or the entire body depending on position of the computer in the control hierarchy.)
- Data from peripherals stored in circulating buffers to permit detection of inconsistencies and subsequent diagnosis.
- Critical sensors could be duplicated; e.g., two vertical gyros.
- Cross-checks possible on related sensors; e.g., integrate tachometers and compare to position transducers.
- Safety computer is a co-processor to coordination computer; accesses bus directly only if coordination computer failure suspected.

DIAGNOSTIC SEQUENCE

1. Put vehicle in safe shutdown configuration.

2. If error detected at coordination level, perform computer tests first. Then go to transducer and I/O check.

3. If error detected at leg level, do transducer and I/O test.

4. If automatic routines fail, provide prompts to maintenance personnel and graphical displays of stored operational data.

DIAGNOSTIC PROCEDURES

- Safety computer functions as a circulating duplex computer checking each other computer in turn via multibus or other connections.

- Transducer function tested by cross-checking of related sensors; e.g., position and force are related.

- Transducer failure cannot be automatically distinguished from A to D or I/O failure. Requires human intervention with auxiliary test instruments and/or direct human sensing (visual, tactile, etc.).

Appendix 3

An Experimental Study of a Binocular Vision System for Rough Terrain Locomotion of a Hexapod Walking Robot

AN EXPERIMENTAL STUDY OF A BINOCULAR VISION SYSTEM FOR ROUGH TERRAIN LOCOMOTION OF A HEXAPOD WALKING ROBOT

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

by

Sheng-Jen Tsai, B.S.E.E., M.S.E.E.

* * * * * *

The Ohio State University

1983

Reading Committee:

Professor F. Ozguner

Professor C.A. Klein

Professor R.B. McGhee

Approved by

R. B. McGhee

Advisor

Department of Electrical Engineering

To My Parents and My Wife

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor Robert B. McGhee and Professor Fusun Ozguner for their constant advice, guidance, and encouragement throughout the course of this work. I would also like to thank Professor Charles A. Klein for his careful reading of the manuscript and valuable suggestions.

I sincerely express my appreciation to Mr. Ronald W. Ventola for his help in many ways during the construction of the interface circuitry. Thanks are due to Mr. Dennis R. Pugh for his discussion in attitude control and active compliance, and his efforts in "taming" the Hexapod from time to time.

I am grateful to Ms. Emily H. Baird and Ms. Joan A. Marn for their excellent work in preparing this manuscript through the word processor.

Finally, I would like to thank my parents for their support and encouragement. Especially, I would like to thank my wife, Shew-Fen, for her patience and understanding throughout my studies.

This work was supported by the Defense Advanced Research Projects Agency under Contracts MDA903-81-C-0138 and MDA903-82-K-0058.

Sheng-Jen Tsai

VITA

June 29, 1953	Born -- Ping-Tung, Taiwan, R.O.C.
June, 1975	B.S., Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.
May, 1978	M.S., Electrical Engineering, State University of New York at Stony Brook, Stony Brook, New York.
1979 - 1980	University Fellow, The Ohio State University, Columbus, Ohio.
1980 - 1981	Graduate Teaching Associate, Department of Electrical Engineering, The Ohio State University, Columbus, Ohio.
1981 - 1983	Graduate Research Associate, Digital Systems Laboratory, The Ohio State University, Columbus, Ohio.

FIELDS OF STUDY

Major Field: Electrical Engineering

Studies in Computer Engineering:	Professors K.J. Breeding, R.B. McGhee, F. Ozguner.
Studies in Control Engineering:	Professors R.E. Fenton, H. Hemami, U. Ozguner.
Studies in Computer and Information Science:	Professors B. Weide, M.T. Liu.
Studies in Mathematics:	Professor J.T. Scheick.

TABLE OF CONTENTS

TABLE OF CONTENTS - continued

TABLE OF CONTENTS - continued

TABLE OF CONTENTS - continued

TABLE OF CONTENTS - continued

LIST OF FIGURES

LIST OF FIGURES - continued

LIST OF FIGURES - continued

LIST OF FIGURES - continued

LIST OF FIGURES - continued

LIST OF FIGURES - continued

LIST OF TABLES

Chapter 1
INTRODUCTION

1.1 Historical Background of Legged Vehicles

It has been observed for some time that biological systems have better mobility and adaptability on rough terrain than conventional wheeled vehicles. People became interested in walking machines for their potential applications in circumstances which are not suitable for wheeled vehicles and environments that are considered hostile to a human being. Examples are remote planet exploration, fire fighting, material handling in nuclear power plants, and mining [1].

Recent advances in the fields of computer technology, automatic control, and a better understanding of animal locomotion from a theoretical point of view have made the realization of legged walking robots possible. Several machines were constructed during the past two decades. Some of these are mentioned below. Each of these machines has its own characteristics and represents a distinct stage of the evolution of walking vehicles.

The Space General Linkage Walker [2] was an eight-legged machine developed in 1960. It was operated by pure mechanical linkage interaction without involving any electronic control circuitry. The drawback of this vehicle was its limited adaptability, owing to the mechanical linkage operation. However, it did demonstrate remarkable off-road mobility.

The General Electric Quadruped [3] was a huge machine weighing about 3000 pounds. It was controlled by a human operator in a master-slave fashion. The machine followed or mimicked the motion of the operator's hands and feet with its front and rear legs respectively. This was a very cumbersome task and demanded a lot of skill. It was concluded that a more efficient way of controlling the machine was necessary.

The Phony Pony [4] was the first walking machine to operate under the control of a special purpose digital computer. The machine was built to show that the joint coordination problem could be solved by a computer.

Compared with the Phony Pony, the OSU Hexapod [5] is a much more sophisticated machine in terms of mechanical structure, electronic hardware, and software algorithms. It was constructed in 1977, and since then a series of experiments have been conducted to study computer hardware and software organization for automatic limb motion coordination. Above all, a supervisory control structure [5] was successfully established. This structure will be examined in the next section. Based on experience with the Hexapod, another new vehicle called the OSU Adaptive Suspension Vehicle (ASV) is now under construction and is scheduled to be completely assembled and tested by 1984.

1.2 The Hierarchical Control Structure of the OSU Hexapod

Although it is a highly desirable goal to automate the OSU Hexapod to the greatest extent possible, every step toward the realization of a fully autonomous machine must be made with consideration for current technology and for available resources. It is believed that a

Follow-The-Leader (FTL) operation is an important milestone toward that goal and is feasible at this time. In such an operation, an operator will control the front feet by specifying footholds for them and the other legs will follow their footprints as the vehicle moves forward, thus only footholds which are assigned by the operator will be used. The relationship between a FTL operation and a completely automatic vehicle can be best illustrated by referring to the control structure in Fig. 1.1.

This structure is called a supervisory control scheme because a human operator is involved in providing high level commands. The function of the control block between the operator and the vehicle is to convert the commands from the former into the physical movements of the latter. Depending on how the control block is implemented, the control hierarchy is divided into four levels.

The lowest level is called the master-slave mode. In this mode, the control block virtually does not exist since the operator has to manually maneuver the machine in all aspects. The operation of the GE Quadruped is of this nature.

The next higher level is the joystick control mode. Here the operator gives such commands as steering and speed through a joystick in a similar way to driving an automobile, and a digital computer furnishes all the low level signals needed to cause the Hexapod to generate the desired behavior [5].

To carry it one step further, in the FTL mode, the operator even does not have to take the wheel. He/she simply indicates a foothold and lets the computer figure out the detailed subcommands to direct the vehicle toward the destination.

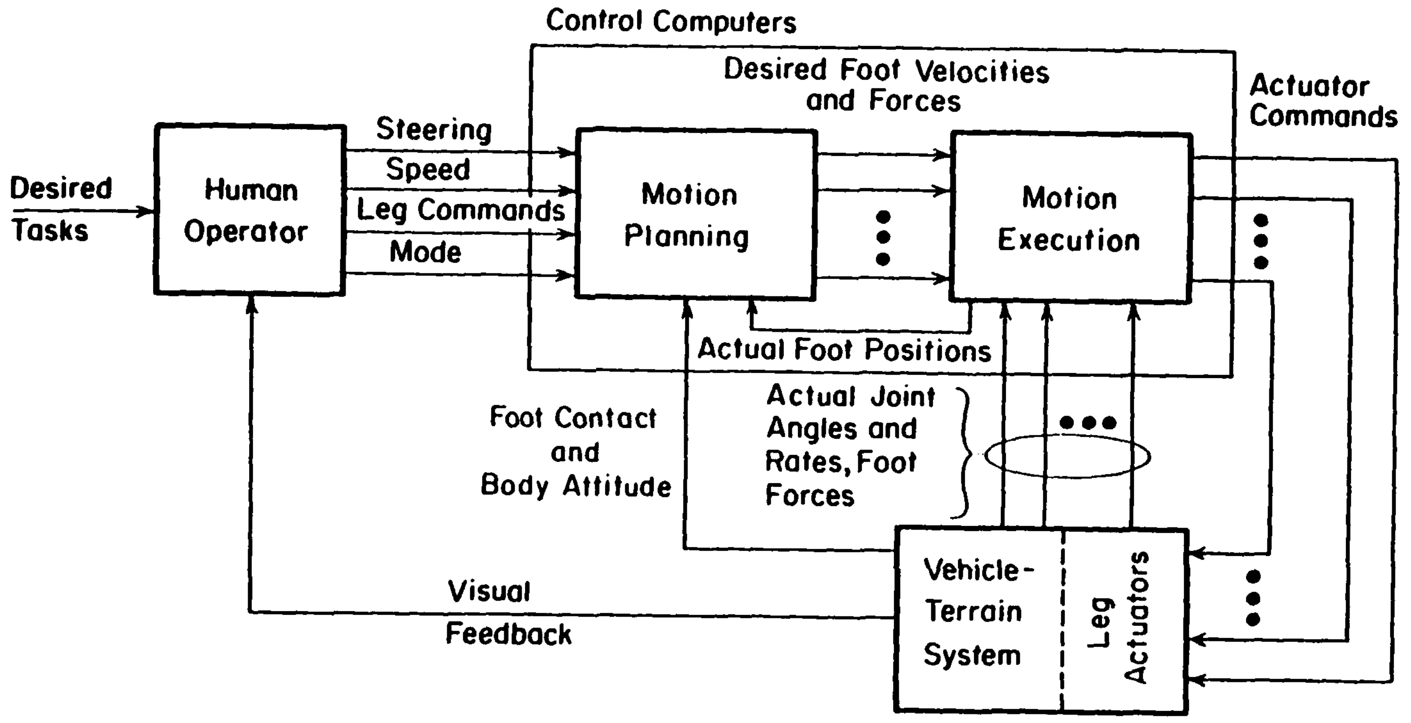


Figure 1.1 Supervisory control scheme used with OSU Hexapod Vehicle [5].

Finally, after sufficient intelligence is built into the machine so that it is able to sense and analyze the environment, and to select a proper route as it moves along, the role of the operator is no longer needed and can be eliminated. As such, the machine becomes self-contained and fully automatic.

The objective of this dissertation is to study and implement the FTL operation. Toward this end, a binocular vision system based on two solid-state TV cameras has been designed and constructed to sense the terrain. In addition, the control of the stepping motions of the Hexapod is conceptualized and a new method of coordinating stepping of successive legs is presented in which only footholds which have been tested by preceding legs in rough-terrain locomotion are used. Finally, all the hardware and software are tested and integrated together to ensure a reliable operational system.

1.3 Organization of the Dissertation

Chapter 2 is a review of related work, including the constituting elements of a vision system, industrial robotics systems that utilize a vision function to aid their operation, and existing mobile vehicles which possess vision capability. Available schemes for range measurements are also discussed. Previous research done on the Hexapod is also covered.

In Chapter 3, the binocular vision system is presented. All components needed to facilitate FTL operation are described. Design and implementation of the hardware interface circuit is explained in detail.

Chapter 4 describes calibration procedures for the constructed vision system. This includes camera calibration and lens calibration as well as the identification of the geometric relationship between cameras and the Hexapod. Coordinate reconstruction based on triangulation is also discussed.

Chapter 5 presents algorithms that direct the Hexapod to generate FTL motion, with an emphasis on the activities occurring at the locomotion-cycle level.

In Chapter 6, modifications that are needed to enable the Hexapod to walk over rough terrain using algorithms developed in Chapter 5 are discussed.

The vision system is evaluated in Chapter 7. Experimental results with the Hexapod walking on even/rough terrain are also shown.

Finally, in Chapter 8, research contributions of this work are summarized and possible extensions are recommended.

Programs related to the vision system and walking algorithms are listed in an appendix.

CHAPTER 2

REVIEW OF PREVIOUS WORK

2.1 Introduction

In recent years, there has been a trend to develop advanced automation techniques to increase industrial productivity and to enhance the well-being of workers by eliminating hard, dangerous and dull jobs [6,7]. Along this direction, robotics has attracted a great deal of attention and induces intensive research efforts [8]. To perform certain human-like tasks, a robot must be able to sense its internal states and its environment. Use of different types of sensors, such as visual sensors, force sensors, acoustic sensors, and temperature sensors have been investigated [9]. Among these, the most important one may be the visual sensor. An old saying goes that a picture is worth a thousand words. While other sensors provide only specific pieces of information, visual sensors have the potential of being used by a robot to construct models of the scene, and to specify the positions, shape, etc. of the objects in an appropriate geometric representation [10]. This is also true for biological systems. Without vision, a human has to rely mainly on contact sensing, which is inefficient due to its point-by-point, trial-and-error nature, and is sometimes dangerous.

One of the central issues in realizing the FTL operation is for the Hexapod to recognize and locate footholds specified by the operator.

This task could typically be well handled by an optical sensing system [11,12,13], specifically, a computer vision system. For this reason, and in recognition of the increasing importance of vision systems as cited above, it was decided to employ a vision system in the hope that, in addition to solving problems related to FTL operation, it would become a beneficial tool for future developments on the Hexapod.

Vision systems are of various types and are usually tailored to suit particular applications [14]. Thus, it is important to know the constituting elements, the role they play, and possible alternatives of a vision system. This subject is addressed in Section 2.2. To provide a clearer picture of the state of the art, several existing systems which utilize vision systems to aid robots in an industrial environment are examined in Section 2.3. Since the Hexapod is a mobile vehicle, it is of interest to understand how a vision system can improve the capabilities of a machine of this class. Five examples are studied in Section 2.4. Range measurement is a crucial part in implementing FTL operation, therefore possible techniques to solve this problem are discussed in Section 2.5. At the end, previous work that has been done on the Hexapod is briefly reviewed in Section 2.6.

2.2 Elements of a Computer Vision System

A typical computer vision system includes an image sensor, an image processor with its own storage memory, and a set of processing algorithms stored in a host computer. A display unit is optional, but is highly desirable in a man-machine interacting environment where an operator monitors the on-going process at the display unit and responds with commands whenever needed. Fig. 2.1 shows a block diagram of a general vision system.

2.2.1 Image Sensors

An image sensor, also known as an image digitizer, converts a natural scene into a numerical representation suitable for digital processing. Four types of widely used imaging devices, image dissectors, vidicon cameras, CCD (Charge-Coupled-Device) cameras and CID (Charge-Injection-Device) cameras are described below.

2.2.1.1 Image Dissectors

A simplified diagram of an image dissector is given in Fig. 2.2. Physically a dissector comprises an evacuated glass envelope which contains a photocathode, an anode with a small aperture, and a signal collecting plate which is directly behind the aperture and collects electrons which flow through the aperture. The operation of a dissector is based on the principle of photoemission, whereby electrons are produced when light fails on a photosensitive material. A light image of the scene is projected onto the translucent photocathode by a lens system. Since the entire scene covers the photocathode, the number of electrons emitted at any time from an elemental area is proportional to the illuminance of that area. These emitted electrons are accelerated toward the anode by the anode voltage. In effect, the light image on the photocathode has been transduced into a corresponding electron density image which moves toward the anode under the influence of the accelerating voltage and the focusing coils. With proper adjustment of the current on the focusing coils, this electron density image is

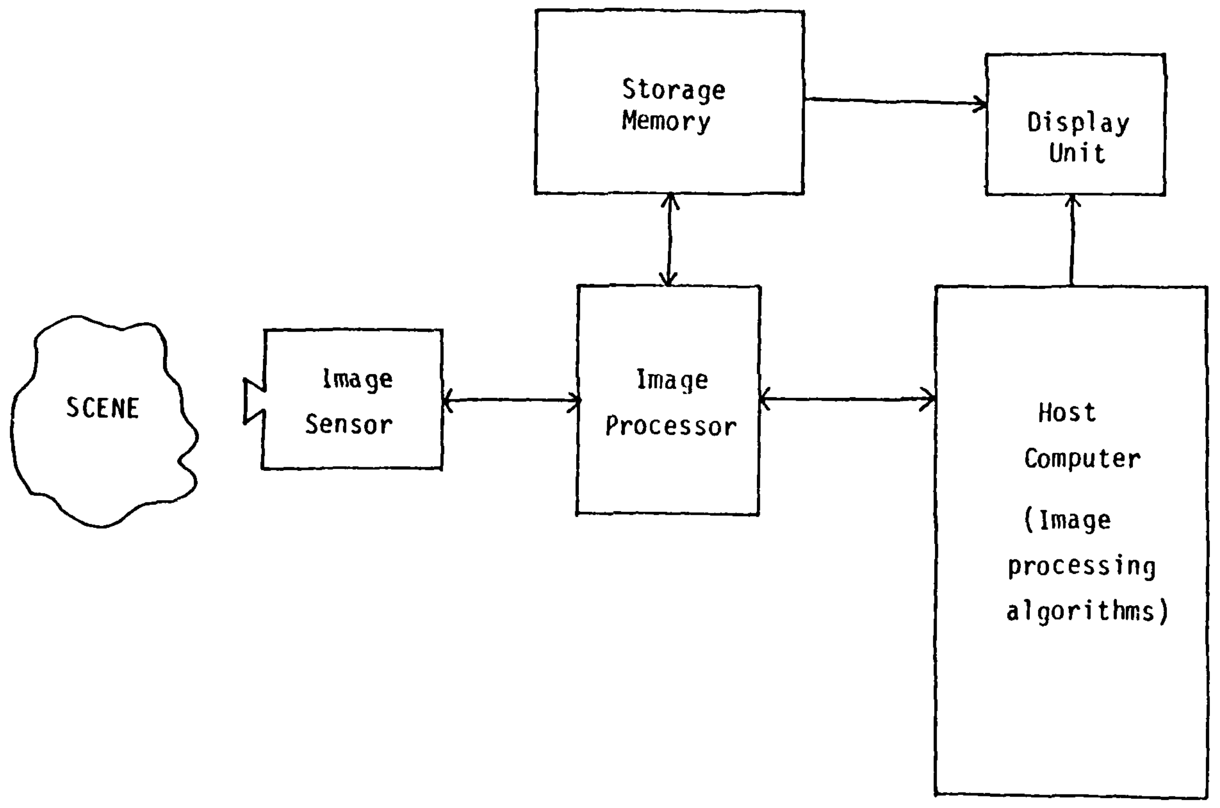


Figure 2.1 Block diagram of a General Vision System.

focused at the plane of the aperture. Obviously those electrons that fall in the area of the aperture continue on to the signal plate and constitute the output current. Since the location of the aperture is fixed, it is necessary to properly control the deflection coils so that the entire image is scanned across the aperture.

2.2.1.2 Vidicon Cameras

Vidicon cameras are popular imaging devices because they are relatively inexpensive and widely available. A vidicon tube is operated under the principle of photoconductivity, i.e., when light falls on a piece of photosensitive material, its conductivity or resistance varies in proportion to the light intensity. Fig. 2.3 shows a simplified drawing of a vidicon tube. Here the target is coated with a transparent conducting film which forms a video signal electrode. On top of this film is deposited a thin photosensitive layer. This layer is so thin that the transversal resistance is high enough to ensure sufficient insulation between neighboring points. Hence, it can be considered as to have a large number of tiny resistive globules whose resistance decreases on illumination. This layer is scanned in a raster fashion by a low speed electron beam from the electron gun at the rear of the tube. The beam deposits electrons on the layer, thus reducing its surface potential. The two surfaces of the target form a capacitor, and the scanning action of the beam produces a capacitive current at the electrode which presents the video signal. The main drawbacks of vidicon cameras are retention and dark current, which is the existing current without illumination.

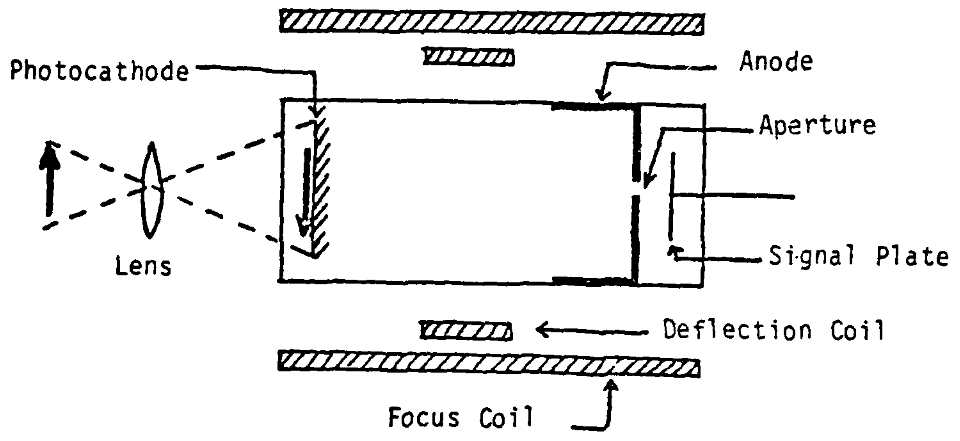


Figure 2.2 An Image Dissector.

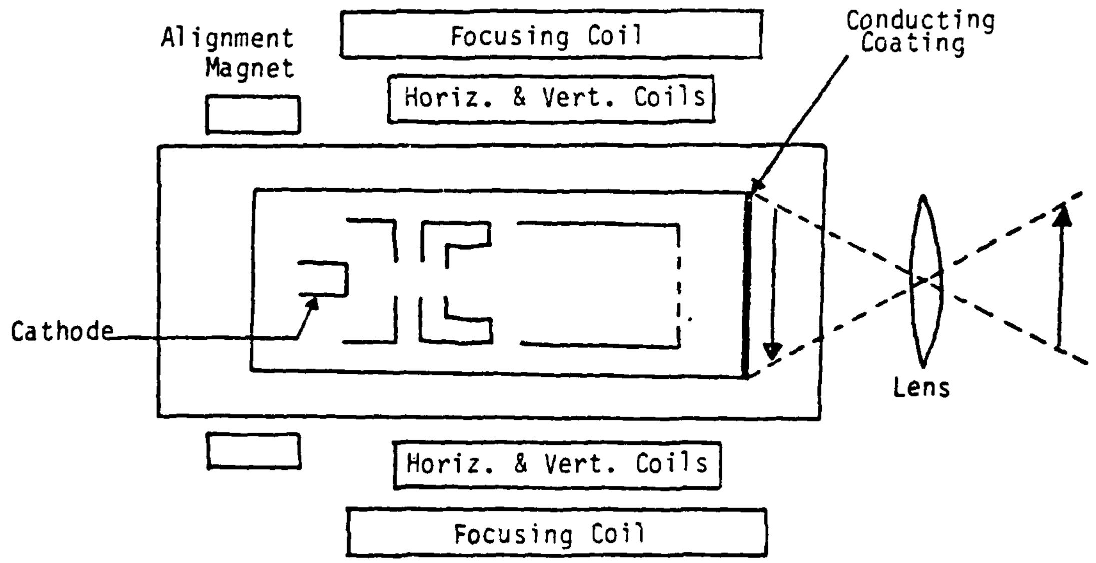


Figure 2.3 A Vidicon Tube.

2.2.1.3 Solid-State Camera

This type of camera offers a number of advantages over vidicon cameras: light weight, small size, low power consumption, low dark current, high sensitivity and lack of persistence. Two different kinds of solid-state cameras are currently available: Charge-Coupled-Devices (CCD) and Charge-Injection-Devices (CID).

For imaging purpose, CCDs can be considered as an array of closely spaced MOS capacitors forming a shift register. Photons incident on the semiconductor generate a series of charges on the CCD array. Charges are transferred to the output by applying a series of clocking pulses to a row of electrodes between the source and the drain.

CIDs resemble CCDs except that during sensing the charge is confined to the image site where it was generated rather than being transferred across the row. The charges are read using an X-Y addressing technique similar to that used in computer memories, then injected into the substrate, and the resulting displacement current is detected as the video signal. Hence, with a properly designed scanning circuitry, each individual pixel of a CID may be randomly accessed, which is not possible for a CCD.

2.2.2 Image Processors

Image data usually is of large volume. Consider a camera which has a 256 by 256 resolution, 8-bit intensity level per image and is operating at 30 frames per second. The resulting data rate is almost 2M bytes per second. Such a large amount of data would take the host computer a long time to process, and thus has been recognized as one of the major bottlenecks for present visually-driven robot tasks [15]. The The role of a hardwired image processor is to free the host computer from the low level, pixel-to-pixel operations so that it can concentrate on executing high-level image processing algorithms. Typical tasks performed by an image processor are as follows.

Image buffering: With sufficient memory space, several frames of image data can be stored by the image processor. This gives the host computer more storage space. It also allows the host computer to process an image from the current frame without losing information contained in the upcoming frames. For example, the VICOM digital image processor manufactured by the VICOM SYSTEM INC. is capable of storing up to 16 consecutive frames [16].

Image reduction: One way to speed up the processing is to reduce the volume of image data. This can be done by either reducing the spatial resolution, say, from 256 x 256 into 32 x 32 as General Motors did in the automation of inspection of transistor ignition assemblies [17], or by reducing the gray-scale image into a binary image [18]. Both techniques result in degradation of image quality; some feature points may be missing in the former method and the latter produces an unsatisfactory picture if the original image has a low-contrast background.

Image preprocessing: Another way to make visual processing faster is to provide fast preprocessing hardware to handle time-consuming but straightforward tasks. Examples are evaluation of image gradient and connectivity analysis. Hitachi has developed a hardware mask matcher that finds the best match between a small (12 x 12) subfield of the image and a reference pattern in the assembly of transistor circuits [19].

Another interesting example is the Selspot System manufactured by the Selcom Company [20]. Tnis system utilizes a two-dimensional PSD (Position Sensitive photo-Detector) to sense the center of the light image and generates output signals which are converted into precise position information. Light sources suitable for the Selspot are light emitting diodes, infrared light sources, or laser sources. It provides X and Y outputs whose amplitudes are directly related to the coordinate of the detected light source. In the most recent version, these outputs are digitized to give 12-bit position data.

The large quantities of image data that need to be processed make parallel computation a highly desirable feature in computer vision systems. Several special architectures have been proposed, including cellular logic image processors, binary image processors, the Preston-Herron processor, the Delft image processor, parallel pattern processors, massively parallel processors, local image processors, adaptive array processors and multiprocessor systems [21,22]. Recently, chips have been developed to execute special computer vision functions at a high rate [23]. It is expected that, as the technologies of VLSI advance, developments will continue to improve chip performance, and other computer vision algorithms will be successfully implemented.

2.2.3 Image Processing Algorithms

The ultimate goal of a vision system is to understand the content of a scene and to extract useful information from it. For example, for a robot to pick up a workpiece from a conveyor, it has to know whether the workpiece is in the scene, if other objects also appear in the picture, and has to distinguish the workpiece to be handled from others, etc. A lot of research and algorithm development has been done in the area of image processing and pattern recognition [24]. In general, scene analysis is done in a hierarchical fashion. First, the whole picture is decomposed into several subregions according to the particular features (edge, hole, corner) that have been detected in each area. Then relationships are established among different areas by some properties (shape, size, color, texture) and the associated regions are labeled with attributes or descriptions. Finally, the recognition of an object is achieved by comparing its attributes with models stored in a library. The foregoing procedures were summarized by Rozenfeld [15] into the four stages of computer vision processes: (1) restoration, (2) segmentation, (3) description, and (4) model matching. A detailed survey of algorithms applicable to each stage is given in [24].

2.3 Vision Systems Applied to Robotics

Four existing systems are described below to exemplify the role of vision systems as applied in industrial environments to enhance the capability of a robot.

2.3.1 The General Motors CONSIGHT System

The CONSIGHT system is designed to transfer parts from a moving conveyor belt to a predetermined location [25]. The conveyor carries randomly positioned parts past a vision station which determines each part's position and orientation on the moving belt. The vision system employes a linear array camera (Reticon RL256C 256 x 1) which images a narrow strip accross the belt perpendicular to the belt's direction of motion. A structured light source, a narrow and intense line of light, is projected across the belt surface (Fig. 2.4). When an object passes into the beam, the light is intercepted before it reaches the belt surface. From the camera's view, there is a dark area on the belt where the part is positioned (Fig. 2.5). The camera data is accumulated, as the part moves along the belt, so that a two-dimensional picture is obtained by combining successive arrays of image. From this picture, properties of the passing part such as area, center of area, X and Y spans, number of holes, and various moment invariants are computed. The part's position is described by the triple (X,Y,A), where the X and Y values are selected as the center of area of the part silhouette and A is the axis of the least moment of inertia of the part silhouette. The robot is a Stanford arm made by Vicarm. The host computer is a PDP-11/45 computer. Since the distance between the camera and the belt is fixed and known, the image analysis involved is a two-dimensional problem instead of a three-dimensional one.

2.3.2 The UNIVISION System

This system is composed of Unimation's PUMA 250/600 robots and Machine Intelligence Corporation's VS-100 vision system, and is controlled by the VAL language [26]. The organization of this system is given in Fig. 2.6. The vision system initially thresholds a gray-level image into a binary image and thereafter works with object silhouettes. It characterizes blobs on the basis of such features as area, perimeter, minimum and maximum radii, and number of holes. The vision system can be "trained" to recognize objects by "showing" sample objects to it. It allows direct connection to GE TN2500 and Reticon LC600C 256 x 1

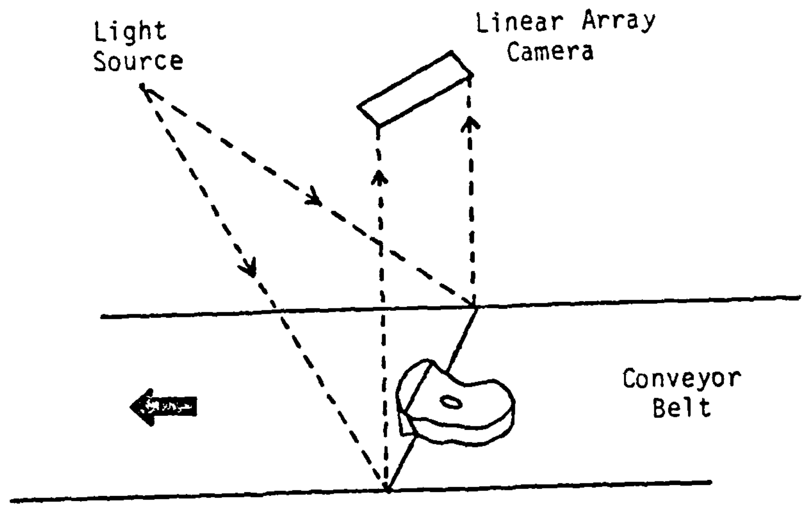


Figure ^.4 Arrangement of the CONSIGHT System.

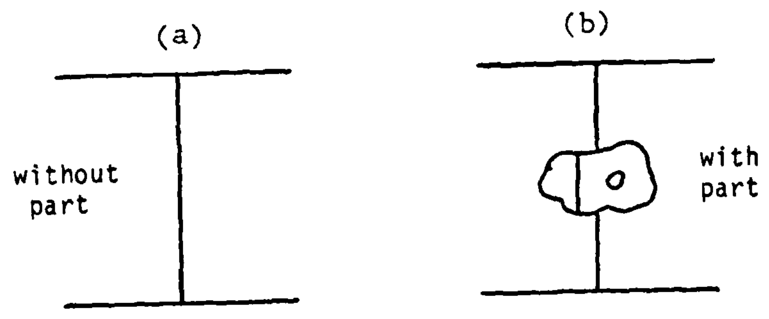


Figure 2.5 Camera's view in the CONSIGHT System.

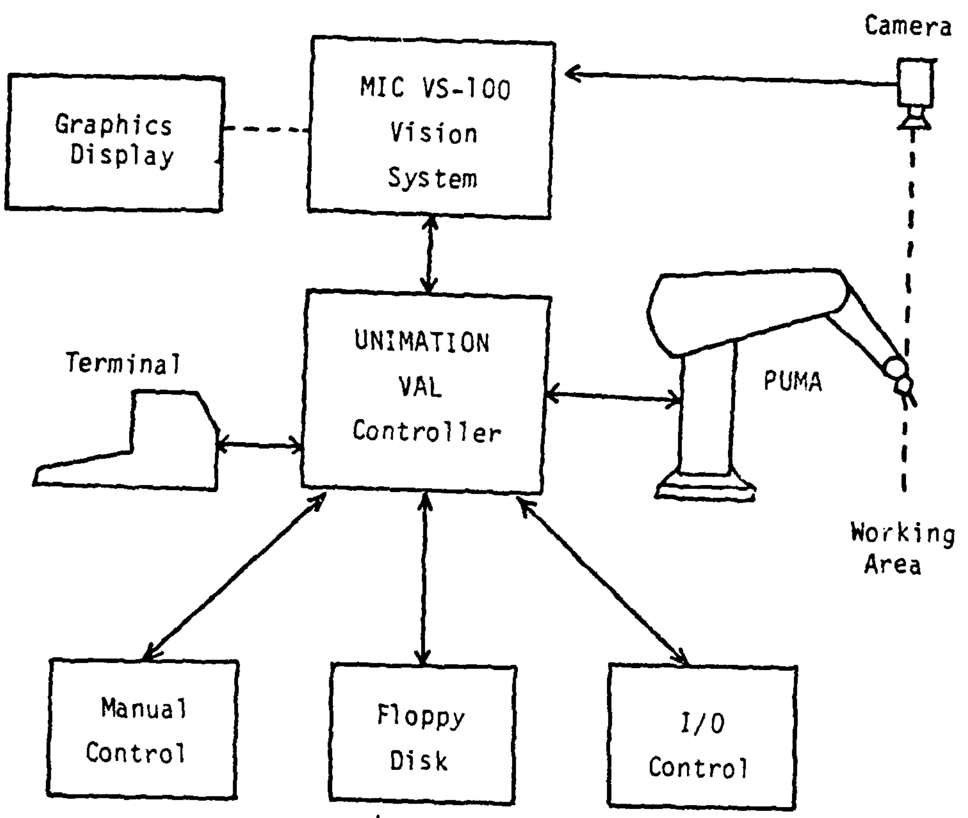


Figure 2.6 The UNIVISION System.

cameras. The locations of visible objects as determined by the vision system are sent to the robot for manipulation.

2.3.3 The Westinghouse Visual Inspection and Industrial Robot Control System

This system consists of a GE TN2200 (128 x 128) camera, a special-purpose preprocessor which performs analog-to-digital image conversion and data storage, a PDP-11/03 microcomputer and a Unimate 500 robot [27]. The structure of the entire system is shown in Fig. 2.7. Vision processing software is developed to perform connectivity analysis, identifying discrete objects in the scene, calculation of the geometric features of the largest object and the object's location and orientation. The vision-robot system is capable of finding, in real time, a workpiece in the field of view with random position and orientation, to identify and inspect it, and to bring it to a specified location with predetermined orientation.

2.3.4 The Robot-Vision System of the University of Rhode Island

This vision system consists of three GE TN200 cameras and their GE PN2110A interfaces [28]. The pixel intensity data and timing signals are routed through a DMA (Direct Memory Access) channel to computer memory. A Grinnell GMR-26 raster graphics display generator, in addition to the ability to generate vectors, rectilinear graphics, alphanumerics and special characters, allows the simultaneous display of up to 4 camera images. One camera is installed on the robot and is used as the primary imaging device to pick up workpieces randomly placed in a bin [29]. The second camera is located at a fixed location and is

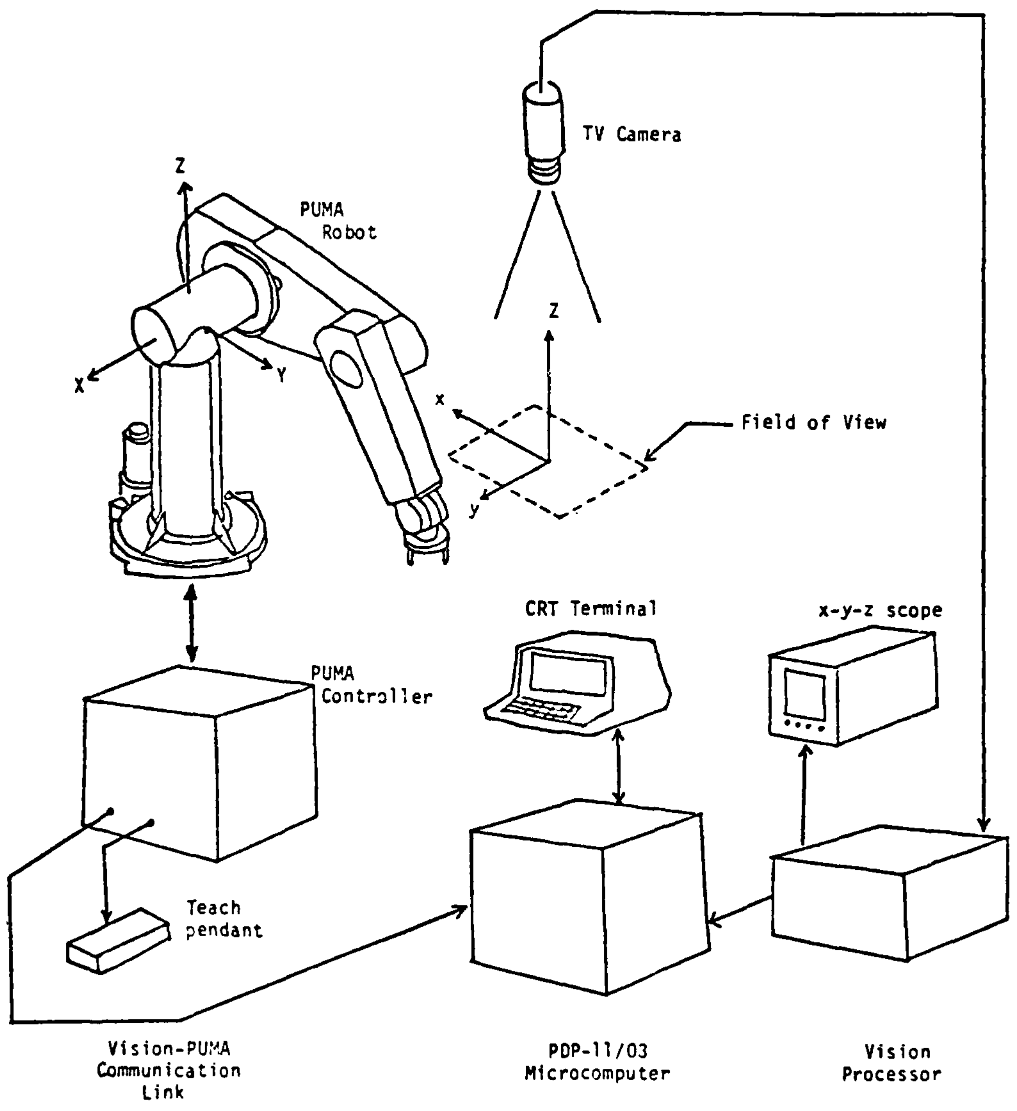


Figure 2.7. The Westinghouse Robot-Vision System.

used to determine the position and orientation of the workpiece held by the robot [30]. The third camera is a spare one. While the three systems described earlier deal mainly with two-dimensional image, URI's system focuses on solving three-dimensional problems. The position and orientation of a workpiece is determined using a stereo vision technique. The workpiece is held by the robot and is presented to the camera. After one picture is taken and analyzed, the arm moves to a different location from where a second picture is acquired. Corresponding feature points in both pictures are identified and triangulation is performed to decide their coordinates. This information is then compared with the model of the workpiece to determine its pose.

2.4 Vision Systems Applied to Mobile Vehicles

The robots employed in the systems mentioned so far all operate at a fixed station which does not move, and in most industrial applications, the working environment of a robot is well structured in the sense that care has been taken to avoid the situations in which a robot runs into unexpected objects by accident. Therefore, the contents of a scene viewed from the vision system may vary frequently, but the surrounding as a whole does not change significantly. However, this is not true for a mobile vehicle. Allowing a machine to move adds another six degrees of freedom to the entire system and certainly make things more complicated. It would be helpful to learn how a vision system is incorporated as part of a problem solver to deal with varying environments. For this reason, five existing mobile vehicles with vision capability will be examined below.

2.4.1 The Hopkins Beast

This machine was built in the early 1960 at the Johns Hopkins University [31]. It resembles a small garbage can on wheels, and its navigation system is based upon sonar measurements that attempt to keep it centered as it moves between two parallel walls. Its eye, a combination of photocells, masks, lenses, and circuits, is designed for detecting electric outlet coverplates since it is a battery-operated system and has to recharge its batteries frequently. Thus as the Beast sees it, the world consists of only two things: coverplates and everything else. As it moves along a hallway, it would suddenly stop, move over to an electric outlet on the wall, stick out its plug, feel for the precise position of the outlet, and plug itself in. That is about all that Beast can do. It is said that its only purpose in "life" is to keep its batteries charged. However, it is a self-contained mobile vehicle since there is no cable or radio link to any computer, power supply or human-operated terminal.

2.4.2 The SRI SHAKEY Robot

This machine possessed a TV camera, tactile and distance sensors, and was controlled by a radio link [31]. It was built in an attempt to investigate research issues in the fields of machine perception and problem solving. The first version of SHAKEY was completed in late 1969. Problems soon flourished after experiments had been carried out. First, its activity was limited to the simplest tasks due to the inadequacy of the computer, which was a XDS-940 with 64,000 24-bit words of slow core memory. Second, the multiple representations of the environment, including (1) a TV picture representation stored and processed in data array, (2) a model of the floor of the room which was segmented into grids and labeled as occupied, unoccupied, and unknown regions, and (3) one that consisted of symbolic axioms of predicate calculus to be used by problem-solving oriented subsystems, were awkward and made it very difficult to update as the world changed. Third, the software developed did not allow the utilization of sensors to check progress, and motion execution was completely open loop, thus a slight error somewhere during execution would result in a crash of the whole system.

The second version was completed by the end of 1971. The computer system was replaced by a more powerful PDP-10 with 200,000 36-bit words of faster core memory. The representation was simplified to a single set of axioms of first order predicate calculus. The software was reorganized to assume a hierarchical structure containing four major levels: Low-Level Action, Intermediate-Level Actions, the STRIPS problem-solving system, and the MACROP executive program. After these modifications, SHAKEY was able to roll about in a laboratory environment consisting of several interconnected rooms populated with wooden blocks.

2.4.3 The Stanford CART Vehicle

The CART was an electrical vehicle remotely controlled over a CB radio link by a PDP-KL10 computer. The vision unit was a black and white TV camera whose picture was broadcast over a UHF channel and digitized by the computer. The goal of this system was to utilize a stereo vision system to guide a robot to move through a cluttered environment, avoid obstacles, navigate to desired locations, and build a description of its environment. With only one camera on the vehicle, stereo vision was accomplished by taking two pictures with the camera moved on a horizontal silder to one of nine different locations. The processing of the stereo pairs was done as follows: First, a special operator found small features with high information content in the first picture. Then, a binary search correlator found the corresponding points in the other picture. After these matched points were identified, the distances to the corresponding points in three-dimensional space were computed using the known camera model. This information was then transformed into a coordinate system approximately aligned with the horizontal surface. A ground surface finder was used to find the ground for portions of the scene, and points which were sufficiently above the ground surface were assumed to lie on objects. This project is currently being pursued by Moravec at Carnegie-Mellon University, where a new vehicle, the CMU Rover, is under construction [11].

2.4.4 The JPL Robot Vehicle

The central component of this system was a scanning Laser Rangefinder (LRF), which was part of a "vision" system that also included two TV cameras and a minicomputer. The purpose of LRF was to provide a means for geometrical, three-dimensional location of objects or surfaces in the neighborhood of the robot vehicle for use as input information to the autonomous control system. The instrument beam, a gallium aluminum arsenide solid-state laser, was directed at specified points or scanned over a specified area under computer control. The detector was a type C31034 photo-multiplier having a gallium aluminum photo surface with special sensitivity to match the laser emission. The range data was measured by using the time-of-flight method. The dual TV cameras were used as a backup system to the LRF and could also provide redundant information.

2.4.5 The RPI Mars Rover

This system employed laser triangulation to compute the distance of an object point. The laser beam was focused onto a multiple-sided rotating mirror which had two degrees of freedom of motion, and then reflected toward a certain direction. Meanwhile, the returned ray from a terrain point was detected by a multi-element linear array of photo-sensors. Since the detectors, laser and mirror were all permanently installed on a mast, their geometric relationships were known. The rotating of the mirror was under computer control. Thus the position of the mirror at every instant was available as well as the angle of the emitted laser ray. The detectors provided the angle of the returned beam. The distance of the corresponding terrain point was computed using this information. Each distance was associated with an elevation and an azimuth angle which indicated the position of the mirror. By incrementing both angles gradually, an area was scanned and a range matrix was constructed. These range data were further processed to characterize the terrain.

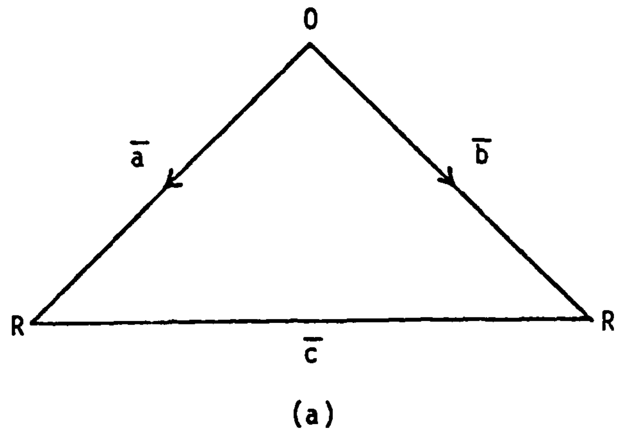
2.5 Techniques for Distance Measurements

During the Follow-The-Leader (FTL) operation, the Hexapod needs to locate a specified foothold and refer it to the current body coordinate system. In essence, Hexapod must be able to measure the distance vector between that foothold and the center of its body. Two types of schemes exist for this purpose; one based on the trigonometry of triangulation and the other based on the time-of-flight of light (or sound). They are discussed below.

2.5.1 Triangulation Methods

In general, three elements are involved in the triangulation method. One is the object point O whose distance is to be measured, and the other two can be both receivers (Fig. 2.8.a) or one transmitter and one receiver (Fig. 2.8.b). Three vectors, denoted as A, B, and C, are defined between them and form a triangle; thus the terminology "triangulation" was originated. Vectors A and B represent optical paths and C is the displacement vector between receivers/transmitter. Once these vectors are known, the range can be calculated by a simple trigonometric formula.

If two receivers are employed, it is called stereo vision and is achieved either with two cameras or one camera at two locations. The receivers are passive devices; however, the illuminating mechanism can be passive or active. For instance, an active laser beam designator is used in this research work to specify a foothold so that the binocular vision system may perceive it, while applications which analyze natural scenes rely on natural lighting only and are considered to have a passive light source. Light sources may also be arranged in a certain way to produce desired lighting conditions. This technique is known as "structured light source method," as demonstrated by the CONSIGHT system [25], and is also considered as active. The most difficult problem in



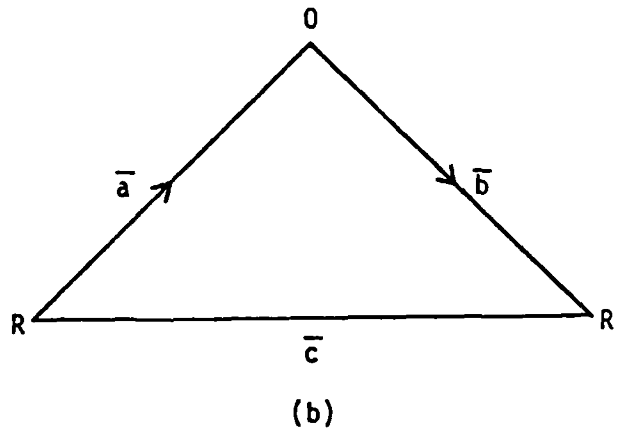


Figure 2.8 Triangulation: (a) with two receivers--stereo vision method;
(b) with one transmitter and and one receiver--light-projecting method.

stereo vision is to find the corresponding points in the two pictures. Although solutions to this problem have been proposed by several researchers [32], there is still need for a more reliable and fast method to perform the computation.

If the combination of a transmitter and a receiver is adopted, it is called the projected-light method. The laser triangulation employed by the RPI Mars Rover falls into this catagory. Other systems project sheets of light onto the scene by means of a slit projector [33], a laser beam diverged by a cylindrical lens [34], or a collimated light beam rotated quickly by a mirror [35]. The correspondence problem is eliminated in this scheme.

Both methods suffer from the following drawbacks: missing data seen by one but not the other, and unsatisfactory accuracy for points that are far away.

2.5.2 Time-Of-Flight Methods

In the time-of-flight method, the range is determined from the time needed for the light to travel from the transmitter to the target and back. If an ultrasonic signal is used instead of light, it can be implemented as a proximity sensor. Such sensors have been designed and built for the OSU Hexapod and have been mounted on each leg to measure the distance from the foot tip to the terrain surface [36]. Laser scanners have been used intensively for measuring range [12,37]. Depending on the measurable range, accuracy, type of laser, and typical applications, laser-based techniques are further classified into interferometric, beam modulation telemetry, and pulse time of flight. A detailed description and comparison of these approaches were given in [38]. The main shortcoming of laser scanner schemes to date is that they are too slow, especially if the target is dark [37]. Such systems also require sophisticated electronics for signal conditioning (modulation, demodulation, filtering, phase detection, etc.) and tend to be more expensive as well as more complex compared to those systems which use triangulation.

2.6 Previous Work on the OSU Hexapod

Before the Hexapod was constructed, Sun [39] investigated the theoretical aspects of legged locomotion systems. Among other things, hexapod gait was studied in depth. Orin [40] conducted a simulation to study the interactive control of the vehicle. The kinematic and dynamic equations were also derived. The Hexapod was built in 1977, with Jaswa [41] handling the mechanical design and instrumentation. Buckett [42] designed, implemented and analyzed the electrically powered joint actuator system. Chao [43] studied the feasibility of a multiprocessor control system for the Hexapod and developed much of the software enabling it to walk under interactive control from an operator. Briggs [44] designed and implemented the data link between the Hexapod and a PDP-11/45 (now 11/70) computer to facilitate data communication. He also installed a vector force sensor on one leg to examine the effect of active compliance. Wahawisan [45] built a pentaprocessor using 5 LSI-11 units to control the Hexapod. Pugh [46] added a force sensor to each leg, installed a gyroscope which provided body attitude information, and devised algorithms utilizing this information to achieve attitude control and active compliance. As a result, the Hexapod was able to walk over rough terrain. Fig. 2.9 shows a picture of the OSU Hexapod before the vision system was installed.

2.7 Summary

This chapter presently a brief review of the state of the art of computer vision systems and their application to industrial robots and mobile vehicles.

A computer vision system, in general, is composed of an image sensor, an image processor, and a set of image-processing algorithms stored in a host computer. Four types of image sensors, image dissectors, vidicon cameras, CCD cameras and CID cameras, were discribed. Solid-state TV cameras, due to various appealing characteristics, are expected to dominate in future applications. Special-purpose image processors are designed to handle the large volume of image data so that the overall operational speed can be upgraded, image data can be properly stored without occupying the memory space of the host computer, and the host computer can concentrate on the execution of high-level algorithms. The purpose of image-processing algorithms is to analyze a scene and to extract useful information from it.

Four existing robot-vision systems, the General Motors CONSIGHT system, the UNIVISION system, the Westinghouse system, and the URI system were examined. In all cases, a vision system has been utilized as a visual sensor to guide a robot to perform certain tasks.

As for mobile vehicles, vision systems are used to survey the nearby environment, to generate a terrain map, and to navigate the vehicle through cluttered surroundings, as examplified by the Hopkin's

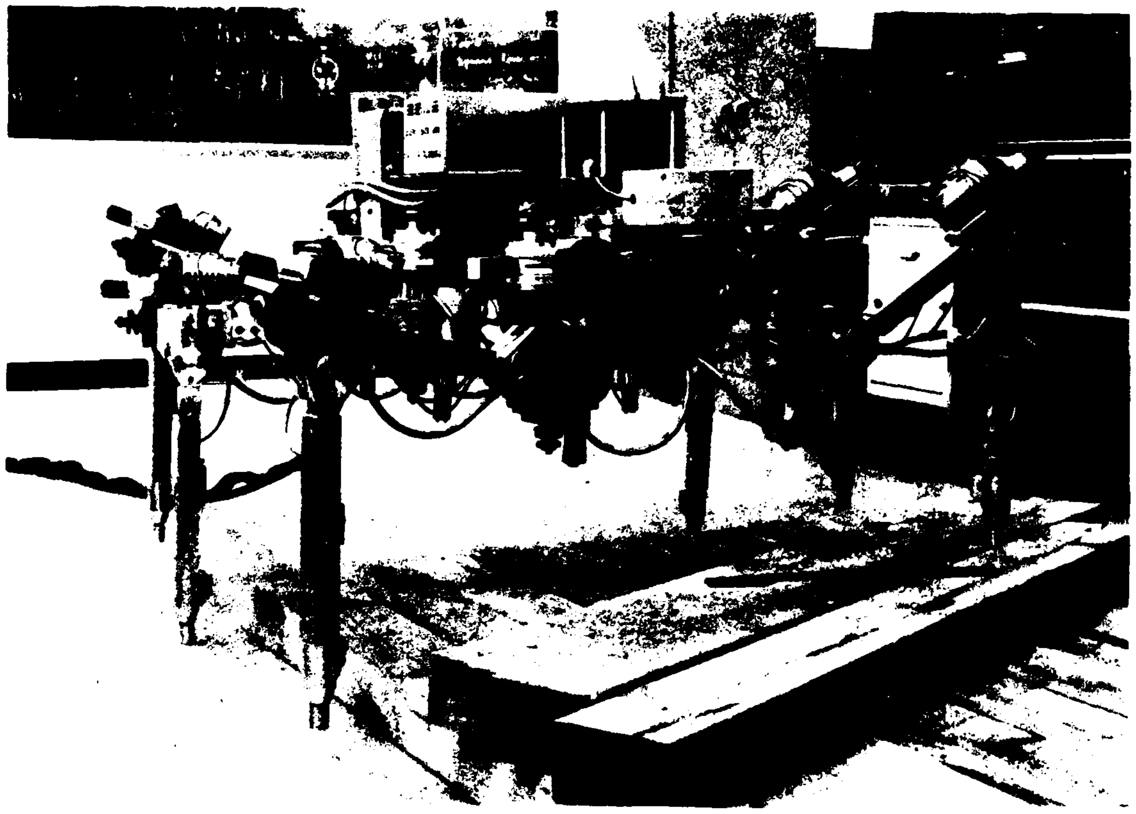


Figure 2.9 Picture of the OSU Hexapod before the Vision System was Installed.

Beast, the SRI SHAKEY, the Stanford Cart, the JPL Rover and the RPI Mars Rover.

Two different schemes for measuring distance were also discussed. The triangulation method is based on the computation of three spatial vectors, while the time-of-flight approach depends on the measurement of the time interval between the outgoing and the returning signals.

Chapter 3

FUNCTIONAL DESCRIPTION OF THE BINOCULAR VISION SYSTEM

3.1 Introduction

The superior off-road mobility of a biological system can be attributed to the following factors:

(1) A body structure which is suitable for locomotion in rough terrain,

(2) A vision system (eyes) that provides environmental information,

(3) A decision-making and coordination capability (brain).

The development of artificial legged machines provides researchers with an experimental body structure similar to that of an animal. However, without the functions of the eyes and the brain, the capability of a walking machine is highly restricted. To improve the versatility of such a vehicle, a vision system, which functions as the eyes, and a computer system, which serves as a brain, are needed. In general, a vision system can be utilized as a terrain sensor and will enable a machine to acquire vital environmental data. Based on this information, computer programs can analyze the terrain, select a safe route, and guide the machine toward a destination. It will be shown in this chapter that with a vision system, one can implement Follow-The-Leader operation for legged locomotion, which is an important step toward the realization of a fully autonomous machine.

In Section 3.2, the Follow-The-Leader control mode is introduced. Problems encountered in realizing it are explored and possible solutions are proposed. Section 3.3 describes the function of each element involved in the proposed vision system. Human interactions are covered in Section 3.4. Operational and testing procedures are given in Section 3.5 so that the normal function of the vision system can be checked. Section 3.6 examines the program which controls the scanning operations of the vision system in detail.

3.2 Follow-The-Leader Control Mode

The successful implementation of a supervisory control scheme [5] relieved a human operator from the cumbersome task of manual coordination of the motions of all joints as exemplified by the G.E. Quadruped Vehicle. Nevertheless, even in its most advanced realization, joystick control operation, human intelligence is still required to provide the desired speed and direction.

The next step toward the goal of a completely autonomous machine is the so-called Follow-The-Leader control operation. In this mode, the operator specifies footholds for the front foot on each side and the others follow in their footprints as the vehicle moves forward. Human intelligence is incorporated to deal with route selection and obstacle avoidance. However, the vehicle figures out for itself the necessary speed, turning rate, direction and movements so that the foot tip touches down at the specified location. This feature is not found in the joystick control mode.

A number of questions arise as an attempt is made to implement Follow-The-Leader control. For instance, one needs to know how a foothold is represented, how it can be generated by an operator, and how it is recognized by the vehicle. After recognizing a foothold, the machine has to know where it is with respect to the current position. Knowing where a foodhold is, it then has to figure out how to move so that one of its front feet is placed down at that point. Also, the nature of the interaction between the operator, the vehicle, and other elements involved, has to be well defined.

Since footholds are actually surface points of a terrain, it is natural to refer them by their spatial coordinates (X,Y,Z). One possible way to accomplish the man-machine interaction as regards to the generation and perception of footholds is to employ a vision system. The operator could indicate a foothold to the vehicle by using an optical medium and the latter would recognize it through an optically sensitive instrument. It is well known that a laser beam is monochromatic, highly directional, and can be collimated into a narrow beam. These characteristics make it an ideal light source for this application. Solid-state TV cameras, on the other hand, have such attractive features as high resolution, high operational speed, small volume, and can be easily interfaced with most digital instruments. Thus it was decided that a laser beam designator be used as the foothold indicator and TV cameras be utilized as the foothold sensing devices.

To know where a foothold is introduces the necessity of measuring its spatial coordinates. This can be done by using the time-of-flight method or by trigonometry. At the time this work was initiated, a dedicated and expensive system based on a time-of-flight laser scanner was under construction by ERIM (Environmental Research Institute of Michigan) to be used with the new vehicle (ASV) to create a terrain map [47]. It was decided that alternatives ought to be explored so that advantages and drawbacks of different approaches would be understood. Hence, the triangulation method was chosen for the research of this work.

Image data acquired by a single camera provides two-dimensional information only. In planning a movement toward a foothold, however, it is necessary to know the three-dimensional spatial coordinates of that point. For applications using cameras, coordinate reconstruction is typically done by triangulation [32]. Furthermore, it can be done by using either one or two cameras. In a single camera system, it is necessary to take at least two pictures from two different views. To achieve this, either the object has to move to a new location as in the URI system [28], or the camera should take a new position as demonstrated by the Stanford Cart [11]. In the vehicle guidance application, the object points are terrain surface points which can not be moved around; thus the former approach is not applicable. The latter method, which requires the camera to move to a second location after one picture is taken, not only is inherently slower than a binocular system due to the time needed to translate the camera, but also demands additional mechanism to relocate the camera. Thus, a binocular system was deemed more desirable and was adopted.

3.3 Description of Components

3.3.1 The Overall System

Five elements are involved in the overall system: a human operator, the Hexapod, a vision system, a laser designator and a PDP-11/70 computer. The block diagram in Fig. 3.1 shows the relationship between these elements.

The responsibility of the human operator is to analyze the terrain and select a proper foothold for the Hexapod. This is achieved by manually adjusting the position and orientation of a laser head so that the laser beam is aimed at a desired surface point. The operator also interacts with the vision system by indicating that a desired foothold is available.

The Hexapod is a six-legged machine built in 1977 as an experimental vehicle for studying locomotion, man-machine interaction, terrain adaptability, fuel efficiency, etc. Each leg has three joints which are driven by individual motors and every joint is equipped with one potentiometer and one tachometer to feed back joint angle and joint rate information, respectively. Commands to the motors are generated by the 11/70. Communication between the Hexapod and the 11/70 is facilitated by a 64-channel Data Link [44]. A vertical gyro on board can provide pitch and roll angles of the Hexapod body. Recently, force sensors have been added to all legs to accomplish attitude and altitude regulation over irregular terrain [46].

The PDP-11/70 computer provides the computional power needed for real-time control of the Hexapod. This is done by executing a motion planning/execution program. A serial link exists between the 11/70 and

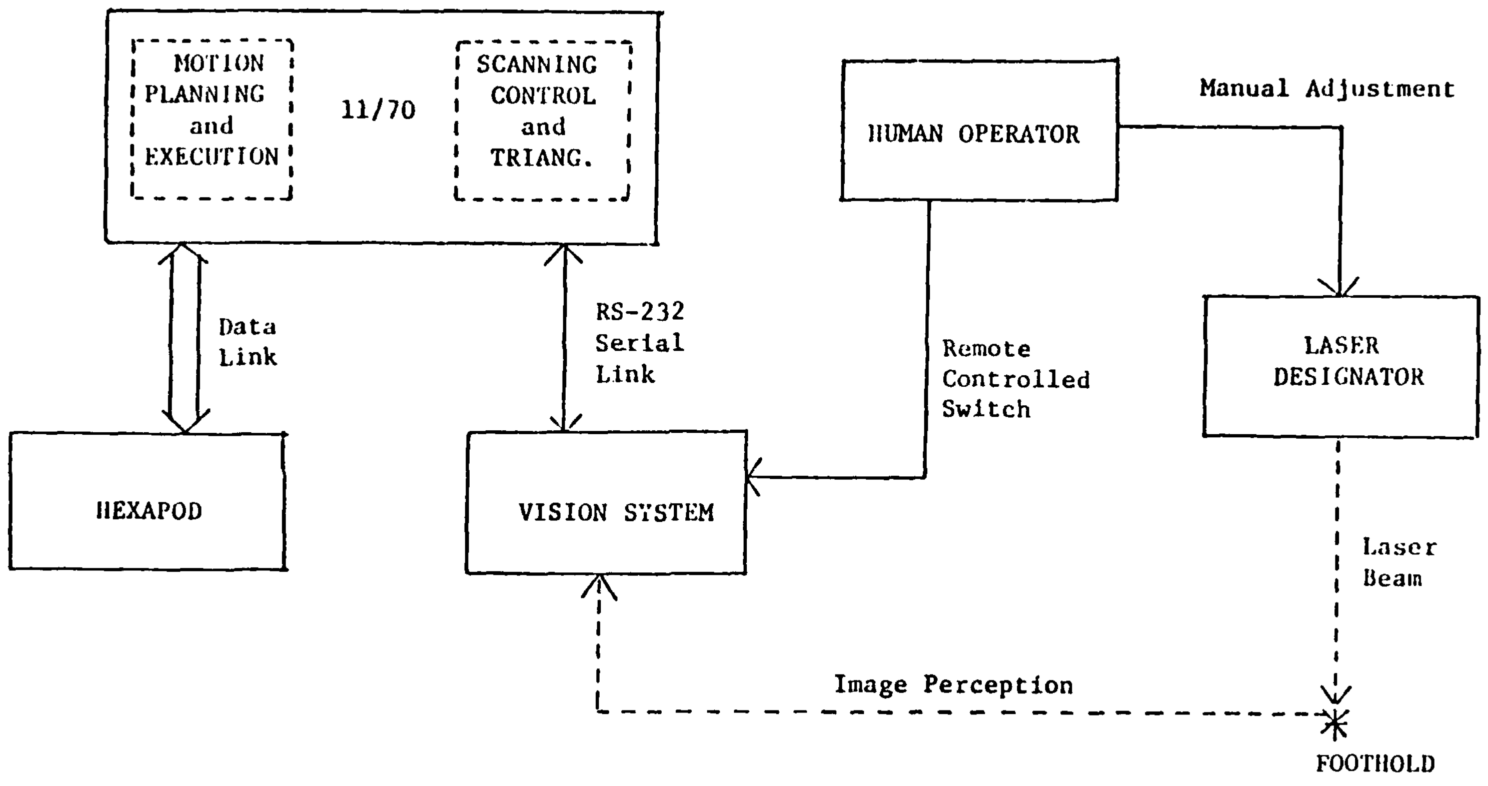


Figure 3.1 Block Diagram of the Overall System for Follow-The-Leader Operations.

the vision system. Through this channel, image data acquired by the vision system is transmitted to the 11/70 for further processing and the 11/70 can send commands to control the scanning operations of the vision system.

The laser designator is used to generate footholds by projecting a laser beam at desired surface points. It includes a 5 mW Hellium-Neon laser head (Ealing Corporation, #25-0860) and a power supply unit (Ealing Corporation, #25-0894), which are shown in Fig. 3.2. When selecting a laser head for this application, size and power level are the two main considerations. Both a 2mW laser head (282 mm long, 31.75 mm in diameter, weight of 0.2 Kg) and a 5 mW one (411 mm long, 44.2 mm in diameter, weight of 0.35 Kg) were tested. While the former has the advantage of size, its lower power level results in lower pixel intensity, and consequently the signal-to-noise ratio suffers. Since it is highly desirable to have a high noise-rejection threshold, a 5 mW unit was chosen. The laser head is mounted on a tripod to make operations convenient.

The vision system consists of two camera heads, two camera controllers, two wide-angle lenses, a special purpose interface circuit, two broad-band optical filters, two filter adaptors and a power supply unit. They are described in the following sections.

3.3.2 Camera Head and Camera Control Unit

The TN 2500 camera head contains a Charge-Injection-Device (CID) image sensor, the logic and drive circuits used to scan the CID, and pre-amplification circuits. The CID makes use of a two-dimensional

Figure 3.2 Laser Head and Associated Power Supply.

array of coupled metal-oxide-silicon (MOS) capacitors to collect and store the photo generated charges. The pixels of the CID are arranged in 244 rows with 248 pixels in a row.

The CCU (Camera Control Unit) consists of a timing circuit board, which generates all of the required digital clocking, gating and synchronization functions, and a signal processor circuit board, which performs all of the required video signal conditioning. It also provides signals for external interface through a 25 pin connector. The block diagram in Fig. 3.3 shows the functional relationship between the camera head, CCU, and external devices, and a picture of the physical devices is given in Fig. 3.4.

Output signals B1 to B8 (Video Bit 1 to Video Bit 8) provide the 8-bit, gray-scaled digital video output, with B1 be the most significant bit and B8 the least significant bit. The elementary clock signal is given at pin ERC (Element Rate Clock). A negative transition of ERC denotes the presentation of new digital video data. In addition to ERC, other timing signals are also available, and each carries different information. A negative transition of EF (Even Field Flag) denotes the start of the even field video presentation, while a negative transition of OF (Odd Field Flag) denotes the start of the odd field video presentation. A high level of SBLNK (Synchronized Blinking) coincides with the presence of digital video output, and a low level coincides with the horizontal or vertical blanking interval. No digital video information is presented when this signal is low. A positive transition of VSYNC (Vertical Sync) denotes the start of each frame when the camera is operated in the sequential mode, and a positive transition of HSYNC

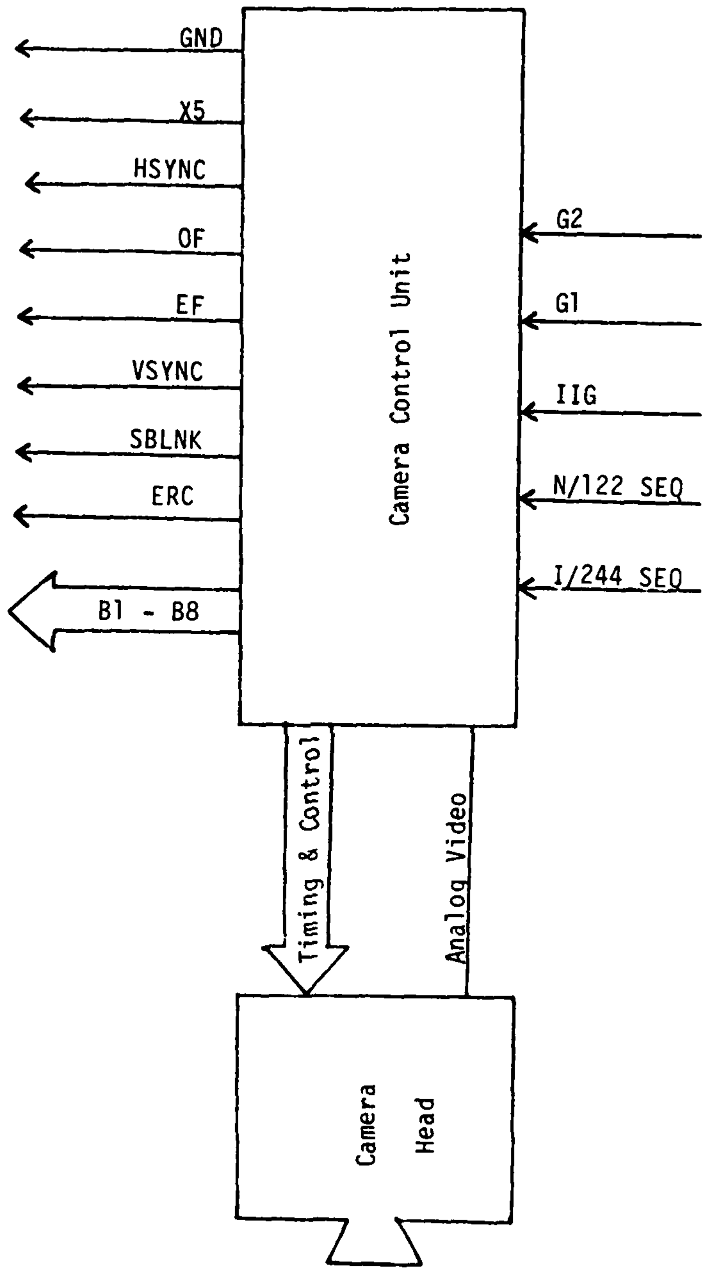


Figure 3.3 Block Diagram Showing Relationships Between Camera Head, Camera Control Unit, and External Interface.

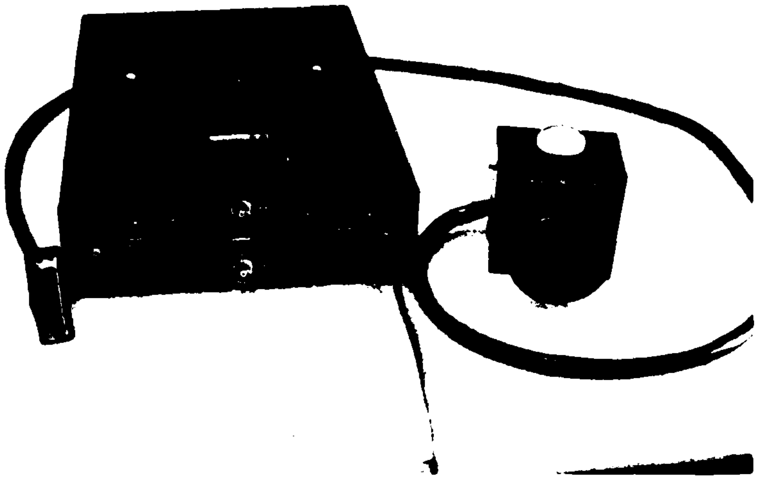


Figure 3.4 Camera Head and Camera Control Unit.

(Horizontal Sync) denotes the start of each horizontal video line. The 5X (5X Clock) is a supplementary clock source whose rate (22.5225 MHz) is 5 times of that of ERC. Timing diagrams Fig. 3.5 and Fig. 3.6 illustrate the relationships between these timing signals.

The input signals are mainly used to regulate the operation of the CCU. A low command at G1 (Video Output Enable) enables the digital video output and a high command forces B1 through B8 to a high impedance state. A low level at G2 (Synchronizing Clock Output Enable) enables the clock outputs. A high level forces all clock outputs into a high impedance state. A high level in input pin I/244SEQ (Interlace or 244 Sequential) forces the camera to operate in the interlaced mode. A low level causes the video to be output in a 244 line fully sequential mode. When the input line IIG (Injection Inhibit Gate) is high, the camera will integrate and inject signal charge once per frame. Changing it to a low level will cause extended integration of the signal charge and inhibit injection. The input pins G1, G2, IIG, and I/244SEQ are controlled externally by using a dip switch. For normal operation, G1, G2 and I/244SEQ are grounded and IIG is pulled high so that the digital video and clock signals are available all the time, the CCU is operating in a sequential mode, and every pixel is injected for each frame.

The individual pixels are read at a rate of 4.5 MHz, which corresponds to one pixel every 222 nsec. Because of this high clock rate, general purpose microprocessors are not fast enough to process intensity data of each pixel, and special purpose hardware is needed. A pixel is cleared as it is read to initiate the reintegration of charge for the next frame, except for the case in which the IIG (Injection Inhibit Gate) is enabled. A pixel will continue to integrate charge without

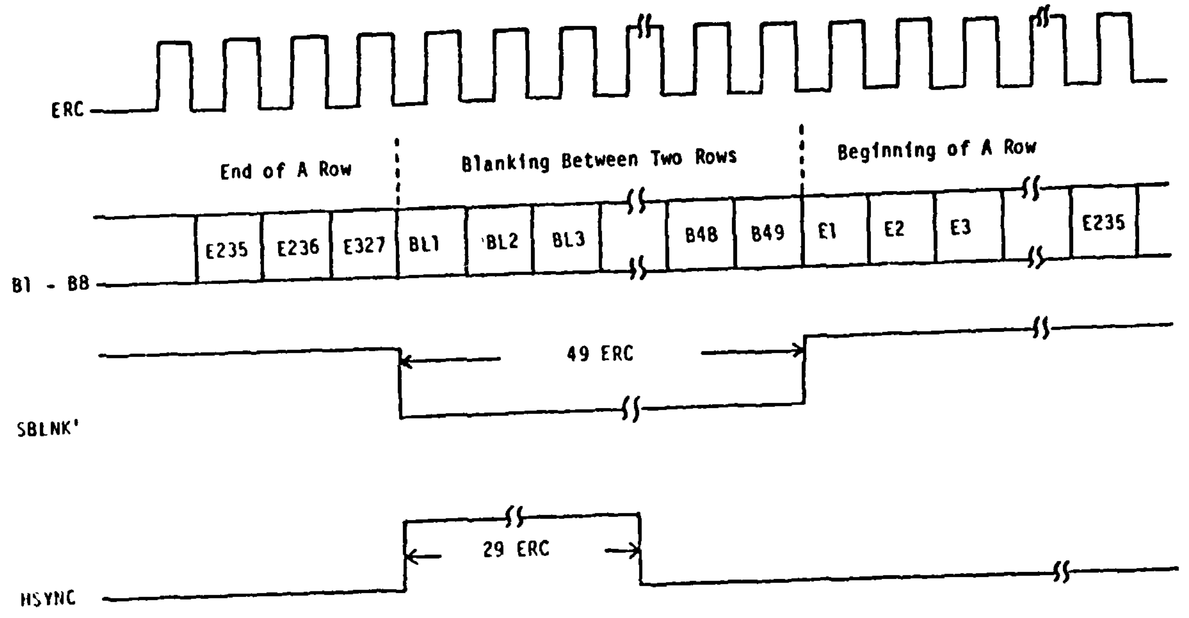


Figure 3.5 Timing Signals Associated with Row Scanning.

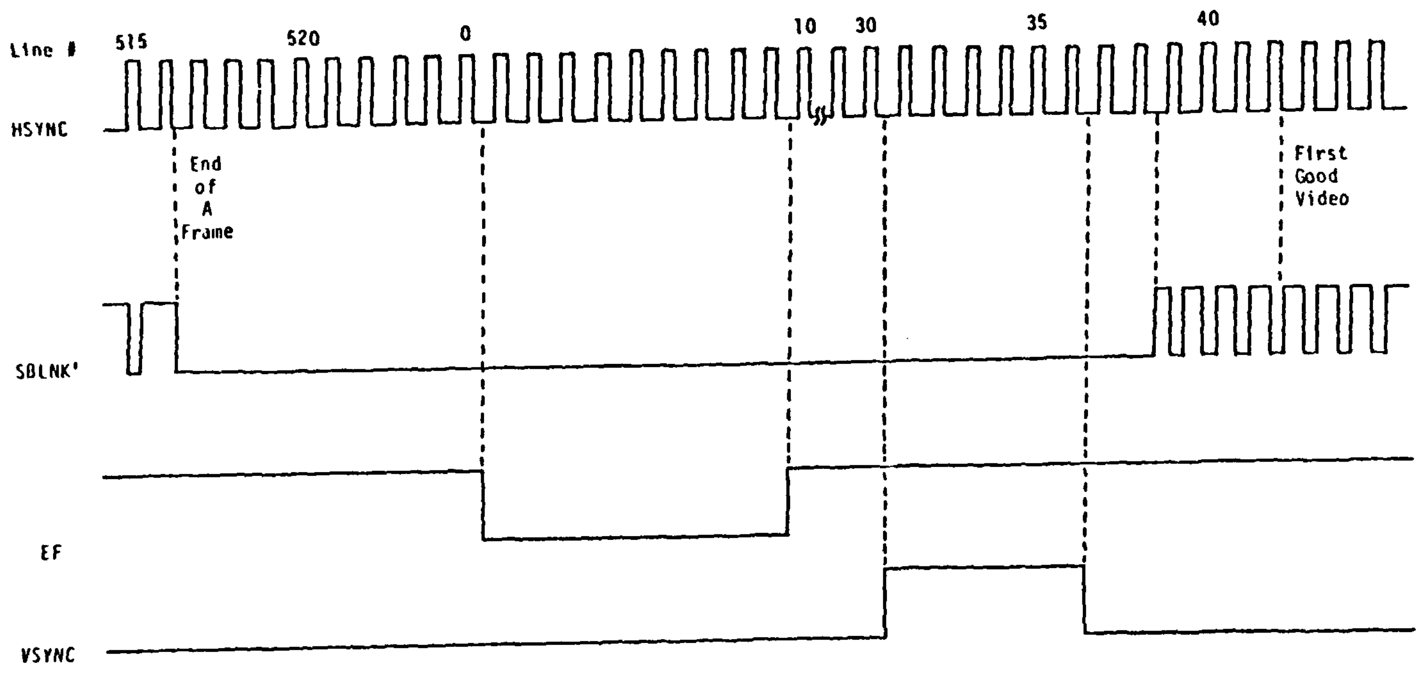


Figure 3.6 Timing Signals Associated with Frame Scanning.

injection as long as the IIG is enabled. This allows a low light image to be accumulated over several frames.

The camera system can provide either an interlaced or sequential raster scan output. In the interlaced mode, each CID image line is read and displayed twice. Odd numbered lines make up field 1 and even numbered lines make up field 2. Subsequently, these odd and even fields are interlaced at a rate of 60 Hz and the resultant frame rate is 30 Hz. This significantly reduces the flicker of the picture on the monitor. In the sequential mode, a frame is composed of only one field. Thus, the field rate equals the frame rate, which is 30 Hz. There is no interlacing and the line information is simply displayed sequentially. This causes a noticeable flicker in the video image.

Although the interlaced scanning mode produces better pictures, it introduces overhead in data processing. For instance, the image data has to be stored or retrieved in an interleaved manner in order to generate a correct picture. Such overhead would not exist in the sequential mode of operation. Therefore, the sequential scanning mode is adopted in this application.

Detailed descriptions of the camera head and the CCU are given in [48].

3.3.3 Camera Locations and Wide-Angle Lens

In this binocular vision system, reconstruction of the three-dimensional coordinates of a foothold is done by triangulation. To achieve this, a foothold must be "seen" by both cameras, i.e.; the foothold must be within the "field of view" of both cameras.

The field of view of a camera system depends on the working distance and its angle of view. A working distance is defined as the distance between the lens center and a focused object along the optical axis. The angle of view, on the other hand, is related to the size of the image sensor and the focal length of the lens. Fig. 3.7 shows the relationship between focal length f, sensor size ℓ, angle of view 2θ and field of view V. It is clear that any light source outside area ACB will not be sensed by the camera. Since

$$\tan\theta = \frac{\ell/2}{f} = \frac{\ell}{2f} \qquad (3.1)$$

tnerefore,

$$2\theta = 2\tan^{-1}\left(\frac{\ell}{2f}\right) \qquad (3.2)$$

It is also obvious that the field of view V at a working distance of d is 2d $\tan\theta$ because

$$\tan\theta = \frac{v/2}{d} = \left(\frac{\ell}{2f}\right) \qquad (3.3)$$

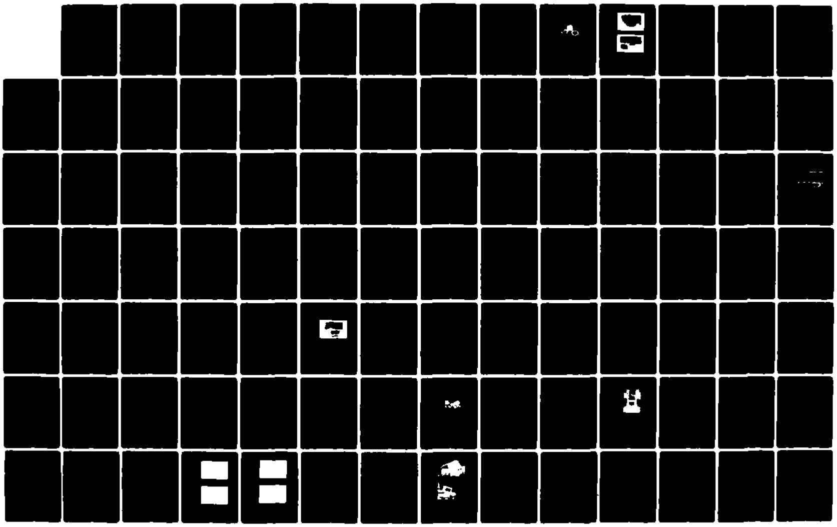
thus

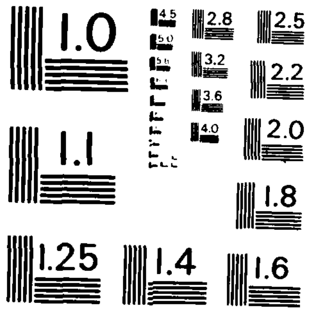
$$V = 2d\tan\theta \qquad (3.4)$$

As the size of an image sensor of the G.E. TN2500 camera is known (8.78 mm by 11.41 mm), only two free parameters are needed to specify a field of view: working distance and focal length of the lens. Since both cameras are to be mounted on the Hexapod and the objects to be

AD-A131 449 AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION(U) OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS R B MCGHEE ET AL. JUL 83 2/4

UNCLASSIFIED MDA903-82-K-0058 F/G 6/4 NL




1.0
4.5
5.0
5.6
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

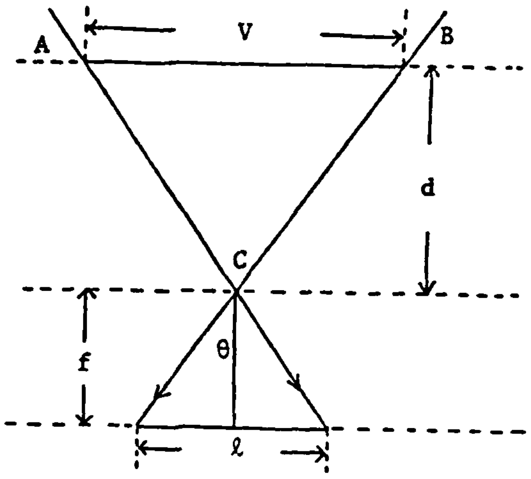


Figure 3.7 Relationship Between Focal Length, Sensor Size, Angle of View, and Field of View.

sensed, footholds, are mostly on the ground surface, the working distance depends on the elevation of the Hexapod with respect to ground and where the cameras are mounted onto the Hexapod body, as shown in Fig. 3.8. The width of the Hexapod is about 150 cm. According to this measurement, a field of view of 2 m is needed for both cameras to cover the area of interest, which consists of points reachable by the front feet in one step. Having decided on a desired field of view, it is now necessary to find out an appropriate camera location and a suitable focal length which, when combined, would give the required field of view.

Refer to Fig. 3.9 and assume that a camera, with an angle of view of 2θ, is located at (X,Y), where $-X0 \leq X \leq X0$ and $0 < Y$. To cover the area of interest, [-X0,X0], it is necessary that $\angle BAC \leq 2\theta$. For the worst case, $\angle BAC=2\theta$, where 2θ is the intersecting angle between vectors $\overline{AB}$ and $\overline{AC}$ and the following equation holds:

$$\overline{AB} \cdot \overline{AC} = \cos 2\theta \cdot |\overline{AB}| \cdot |\overline{AC}| \tag{3.5}$$

where

$$\overline{AB} = (-X0-X,\ -Y) \text{ and } \overline{AC} = (X0-X,\ -Y).$$

Substituting $\overline{AB}$ and $\overline{AC}$ into Eq. (3.5) results in

$$(-X0-X,\ -Y) \cdot (X0-X,\ -Y) = \cos 2\theta \cdot \sqrt{(X + X0)^2 + Y^2} \cdot \sqrt{(X - X0)^2 + Y^2} \tag{3.6}$$

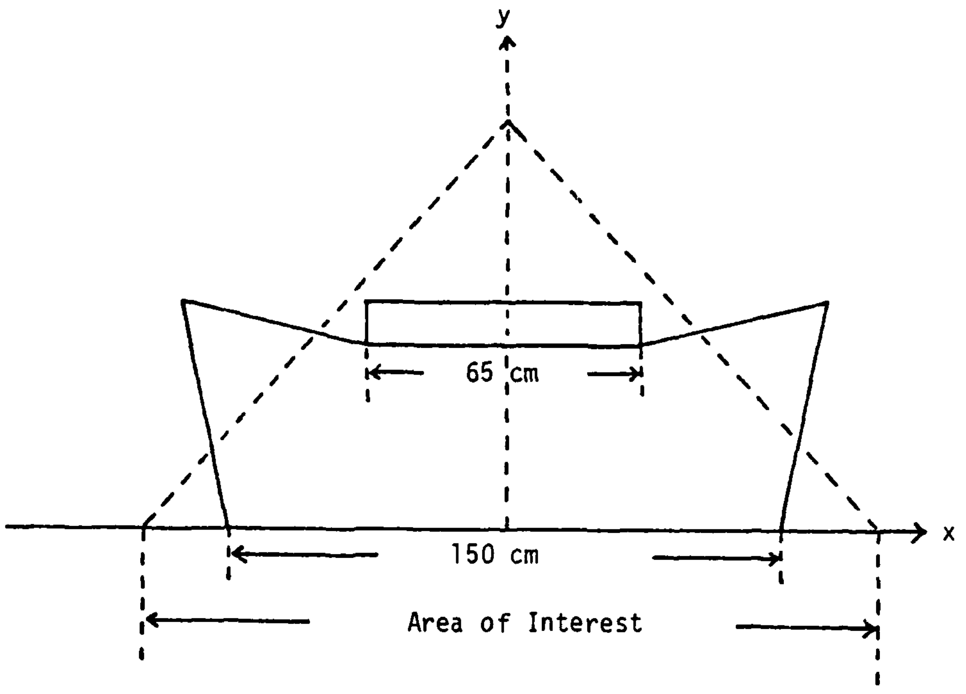


Figure 3.8 Hexapod Body versus Area of Interest.

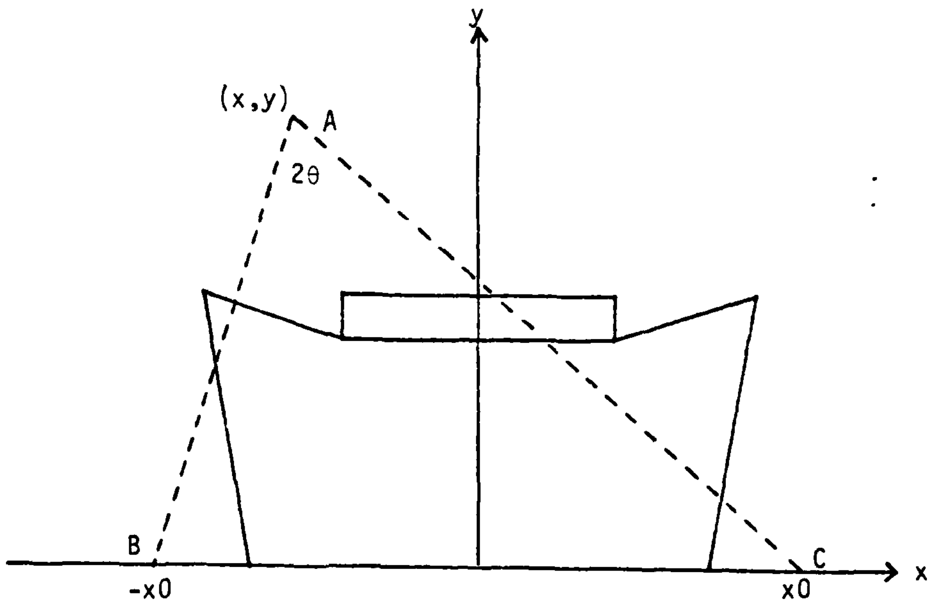


Figure 3.9 Locations Giving the Required Field of View.

Multiplying the left hand side terms out gives

$$Y^2 + X^2 - XO^2 = \cos 2\theta \cdot \sqrt{(X + XO)^2 + Y^2} \cdot \sqrt{(X - XO)^2 + Y^2} \tag{3.7}$$

Employing Eq. (3.1) gives

$$Y^2 + X^2 - XO^2 = \cos 2 \cdot [\tan^{-1}(\frac{\ell}{2f})] \cdot \sqrt{(X + XO)^2 + Y^2} \cdot \sqrt{(X - XO)^2 + Y^2} \tag{3.8}$$

Thus, given a sensor of size ℓ and a lens of focal length f, there is a set of points on the X-Y plane which provides the desired field of view. Fig. 3.10 shows curves of these set of points for a 9 mm by 9 mm sensor for focal lengths of 2, 4, 6, 8, and 10 mm, respectively. Theoretically, there is no restriction on the value of Y; i.e. the height of the camera mounting. However, from a practical point of view, it is desirable to keep the camera position as low as possible to avoid mechanical instability. The Hexapod itself is about 75 cm high and a decision was made to limit camera height to be less than 150 cm to maintain structural stability. These two considerations imply a focal length between 5 mm and 7 mm. Consequently, two 6.5 mm wide-angle lenses were purchased.

Although a wide-angle lens provides a larger field of view, it also introduces considerable distortion, and lens calibration is required to compensate for the distortion. This topic is discussed in Section 4.4.

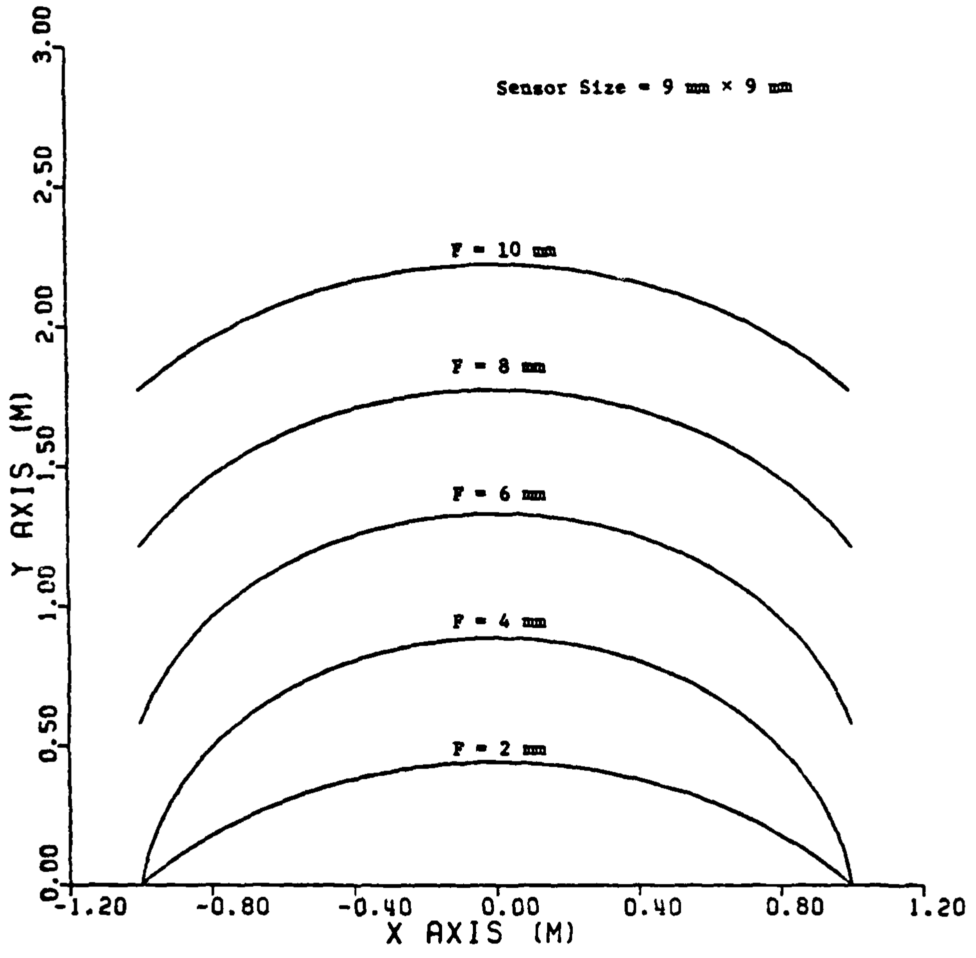


Figure 3.10 Camera Locations Which Give the Desired Field of View as a Function of Focal Length.

3.3.4 Optical Filters

Since the interface circuit is designed to detect only one target point, any ambient light source whose intensity is higher than the threshold and whose location is scanned earlier than the desired spot will be recognized as a target, thus producing a wrong result.

Internal thermal noise of a CID can generally be handled by choosing a proper threshold. External noise, such as sunlight or fluorescent lamp, can be so strong that it is impossible to exclude them by a higher threshold because the desired object source may also be blocked out.

One alternative to reject ambient light is to utilize optical filters. Since the desired light source in this application is a Helium-Neon laser beam which is monochromatic and has a wavelength of 632.8 nm, it is expected that a narrow-band filter would allow the laser beam to reach the camera while filtering out most of the unwanted light sources. One such filter (Oriel Corporation, #5308) with a bandwidth of 10 nm was attached in front of the camera lens by using a specially designed adaptor. Although it blocked out most of the ambient light, the following problem was introduced. The passing wavelength was specified as 632.8 nm. However, this is true only if the incident light is normal to the filter surface. As the incident light deviates from the normal direction, the effective passing wavelength becomes shorter. This phenomenon is described by the following equation [49]:

$$\frac{\lambda_a}{\lambda_n} = \frac{\sqrt{n^2 - \sin^2 a}}{n} \qquad (3.9)$$

where

a = angle of incidence,

λ_a = peak wavelength at angle a,

λ_n = peak wavelength at normal incidence,

n = incidence angle shift factor. (n has a value of 2 for the filters that are used in this system.)

As a result, the effective field of view is sharply reduced.

The wide-angle lens used has a 90 degree angle of view, with 45 degrees on each side of the normal direction. If the reflected light is normal to the filter surface, the incident angle is zero and the effective peak wavelength, according to Eq. (3.9), is λ_n (632.8 nm). However, the incident angle is 45° in the worst case, which produces an effective peak wavelength λ_a of 580 nm for n equals to 2. This calls for a bandpass filter which passes light of wavelength between 632.8 nm to 580 nm. No commercially available filter has exactly this desired characteristic. To compromise, a broad-band filter with a peak wavelength at 600 nm and a bandwidth of 70 nm (Oriel Corporation, #5760) was used. As one may expect, the noise rejection capability is inferior to the narrowband counterpart; nevertheless, the decrease of the field of view is less severe. A picture showing the broad-band filter and a adaptor which mounts the filter onto the lens is provided in Fig. 3.11.

Another way which can effectively reject ambient light is to reduce the aperture of the lens to a minimum. Fig. 3.12.a shows a picture taken with the aperture set to f/16. Fig. 3.12.b shows a picture of the same scene taken with the aperture set to f/5.6. It is obvious that the change in aperture significantly rejects ambient light. The

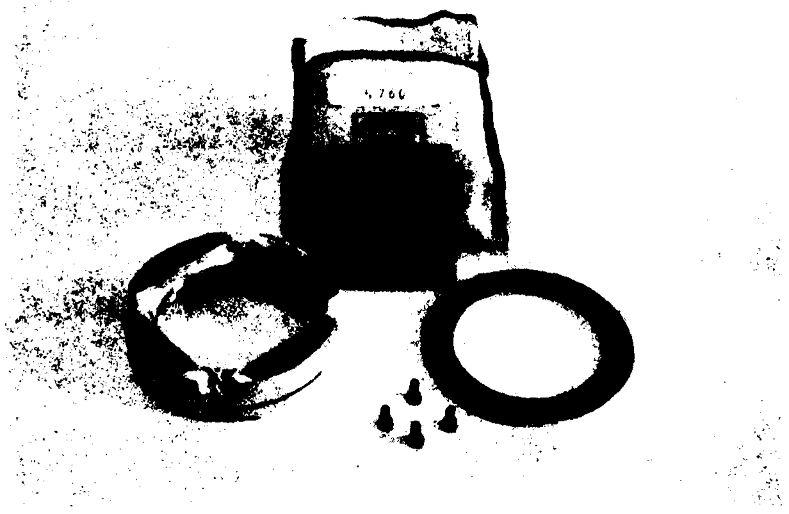
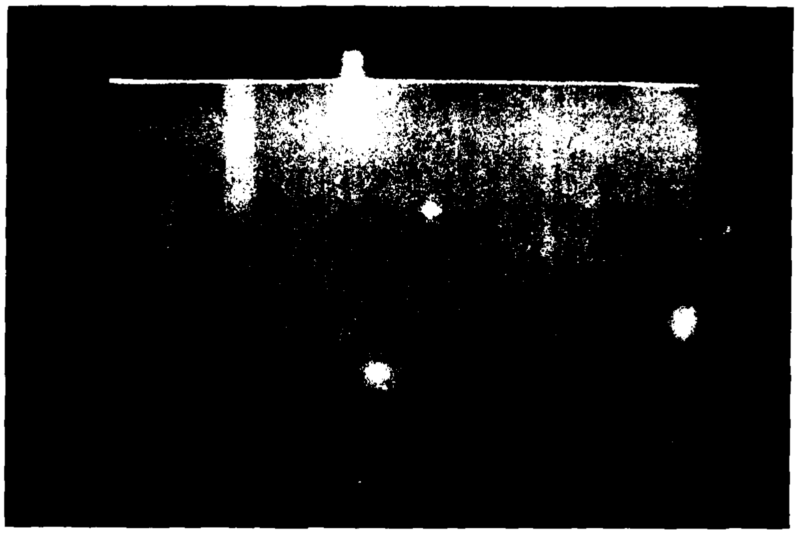
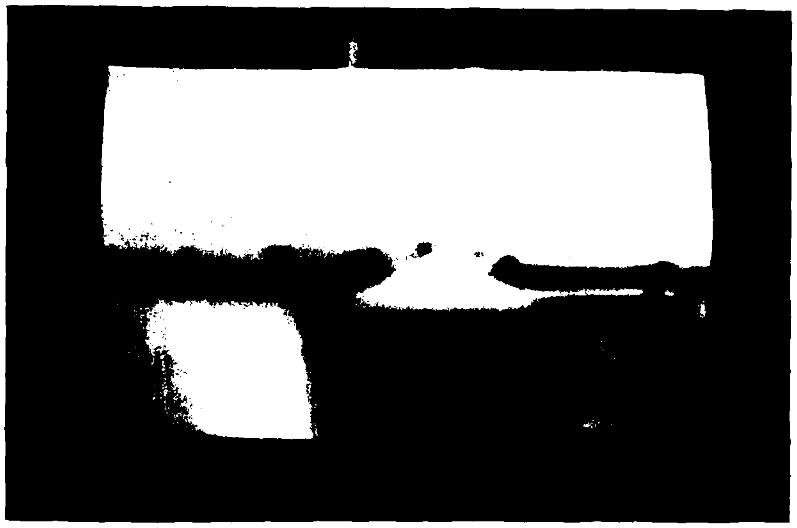


Figure 3.11 Optical Filter and the Associated Adaptor.



(a) f/16



(b) f/5.6

Figure 3.12 Pictures of the Same Scene Taken with Different Apertures.

reflected laser beam is also attenuated sharply, and the size of the image of the laser beam is reduced into a very tiny spot. It is likely that the nonlinearity of the internal electronic circuits of the camera head may have improved the signal-to-noise ratio. The effect of the aperture thus is similar to an analog threshold device. If an extremely strong light source exists in the field of view, however, it would not be completely rejected even if the aperture is set to a minimum. Such situations should be avoided for the vision system to function properly.

After testing, it was concluded that the most satisfactory way of rejecting ambient noise while maintaining sufficient reflected laser intensity is to use a minimum aperture (f/16) without optical filters.

3.3.5 Power Supply Unit

The power supply unit provides the interface circuit board with the necessary +5 and ±12 volt sources. The ±5V module can provide up to 3 amperes, while the ±12V module has a current capability of 1 ampere.

3.3.6 Special Purpose Interface Circuit

The primary function of the interface circuit is to receive pixel intensity data and timing signals from both of the camera systems, determine the image coordinates of an object point on each camera frame, and then transmit these coordinates to the 11/70 computer through an RS-232 serial link. Fig. 3.13 shows the block diagram of the interface circuit. It contains an interface controller, two image coordinate decoders, and an optical isolation unit. Functions and implementations of these circuits are described below.

3.3.6.1 Interface Controller

The interface controller is composed of two parts, an image decoder controller and a UAR/T controller. The former regulates the activities and maintains the synchronization of the two image coordinate decoders. The latter supervises the communication between the interface circuit and the 11/70 computer.

3.3.6.2 Image Decoder Controller

As shown in Fig. 3.13, each image coordinate decoder receives timing signals from its associated CCU to perform image detection. Since, functionally speaking, the two CCU's are totally independent, the operation of the image decoders would be asynchronous to each other. This makes the acquisition of image coordinates difficult, as explained in the following example.

Fig. 3.14 shows a relative timing diagram of the two CCU's in terms of the detection of an image within a frame. Image decoder 1 starts searching at the beginning of a frame, marked as "a", detects an image at instant b and ends up at c, where a new frame starts and the same event is repeated again. Similar activity takes place in image decoder 2 between d and f. The question arises of when image coordinates should be acquired for future processing. While (X1,Y1) is available throughout period [b,c], (X2,Y2) is not ready until e. At this point, however, (X1,Y1) is no longer valid because a new cycle has already begun.

This problem can be eliminated by introducing two synchronizing signals, START and COMPLETE. Fig. 3.15 shows the timing diagram in

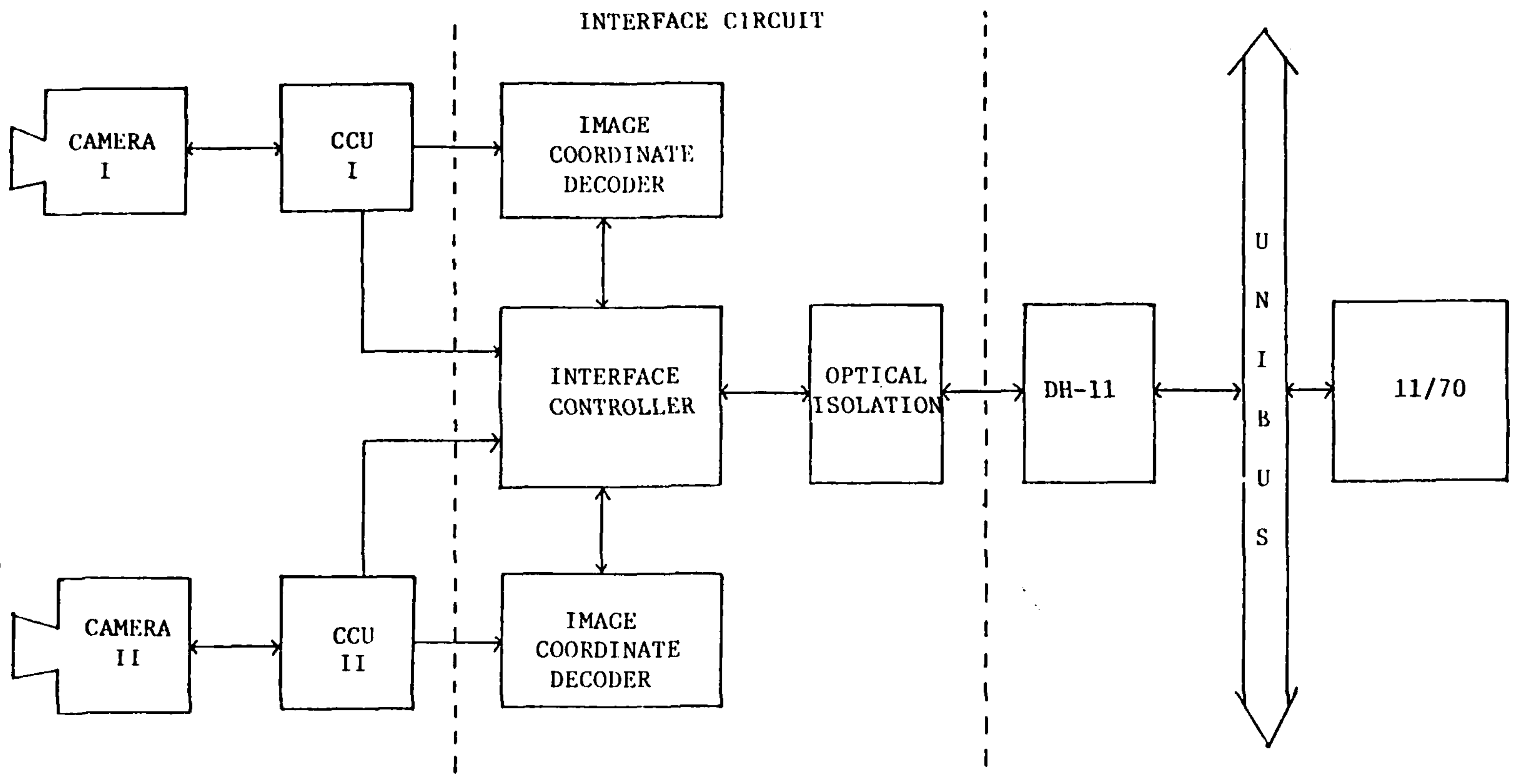


Figure 3.13 Organization of the Binocular Vision System.

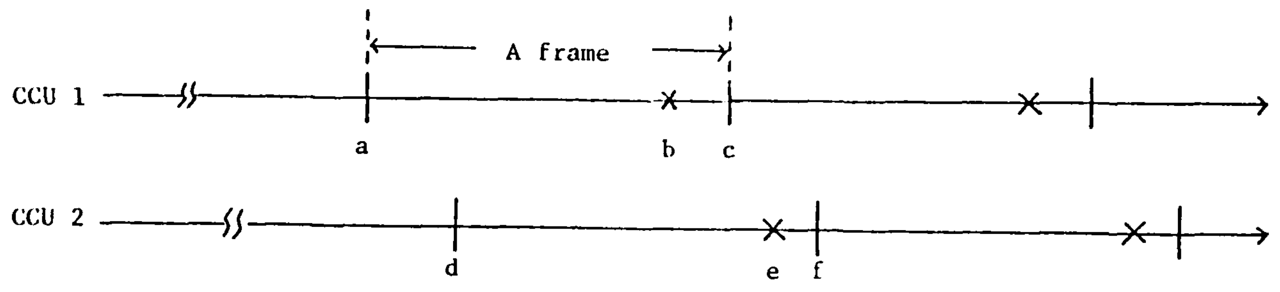


Figure 3.14 Timing Diagram of Two Asynchronous CCU's.

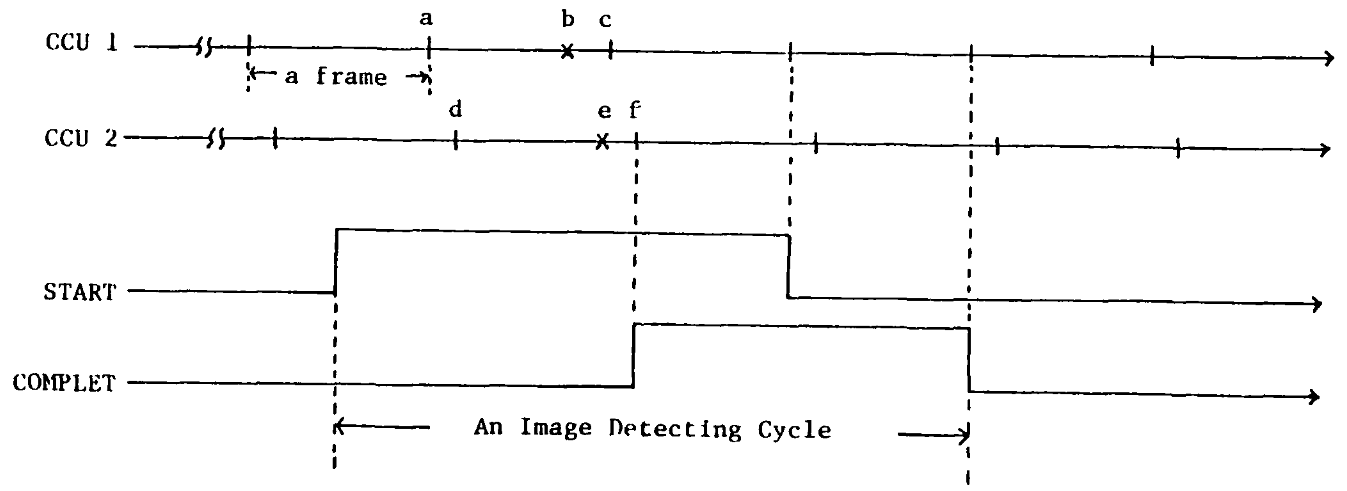


Figure 3.15 Timing Diagram of Two Synchronized CCU's with START and COMPLETE Signals.

which START and COMPLETE are incorporated. Notice that an image detecting cycle now takes more than one frame. The next paragraph describes how the START and COMPLETE signals are used to regulate the function of the image coordinate decoders and how they are generated.

To synchronize the image decoder controller, each image decoder is assigned four states, as shown in Fig. 3.16. Here, the state variables of decoder 1 are Q1 and Q0, and those of decoder 2 are Q3 and Q2, respectively. After POI (Power-On Initialization) or being reset externally, an image decoder is in state 00. It stays at this state as long as START is low. When START becomes high, it enters the image searching state 01 upon the leading edge of the next coming clock signal (EF) which indicates the start of a new frame. It spends exactly one frame period at state 01 performing image detection then moves into state 11. It will wait here until COMPLETE becomes high, denoting that both decoders have reached state 11 of their own. It then enters state 10 during which START will be cleared. Finally, it goes back to state 00 and waits here for START to go high to begin another cycle.

The above state diagram was realized in Fig. 3.17 by using four 74112 J-K flip-flops and a number of gates. The clock signal which triggers the transition from one state to another is the EF (Even Field) signal from CCU. An EF signal always advances a new frame, thus conveying the vital timing information.

The START signal, which initiates an image detecting cycle, can be set either manually through a switch S1 on the interface board or under the control of the 11/70, which uses bit RD1 to trigger the operation when switch S4 is set to a '1' (scanning mode) and both decoders are at state '00'. Signal RD1 is the least significant bit of the byte of data

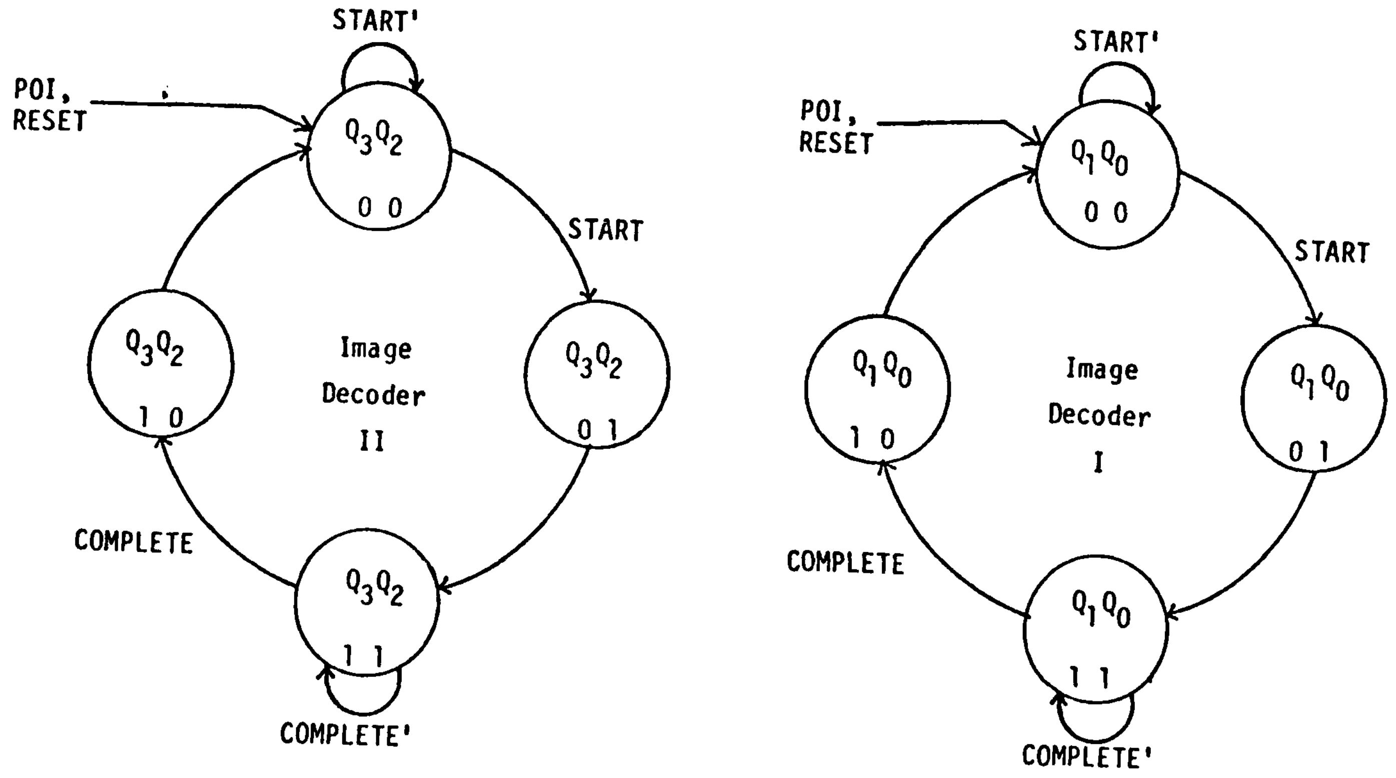


Figure 3.16 State Diagram of the Image Decoder Controller.

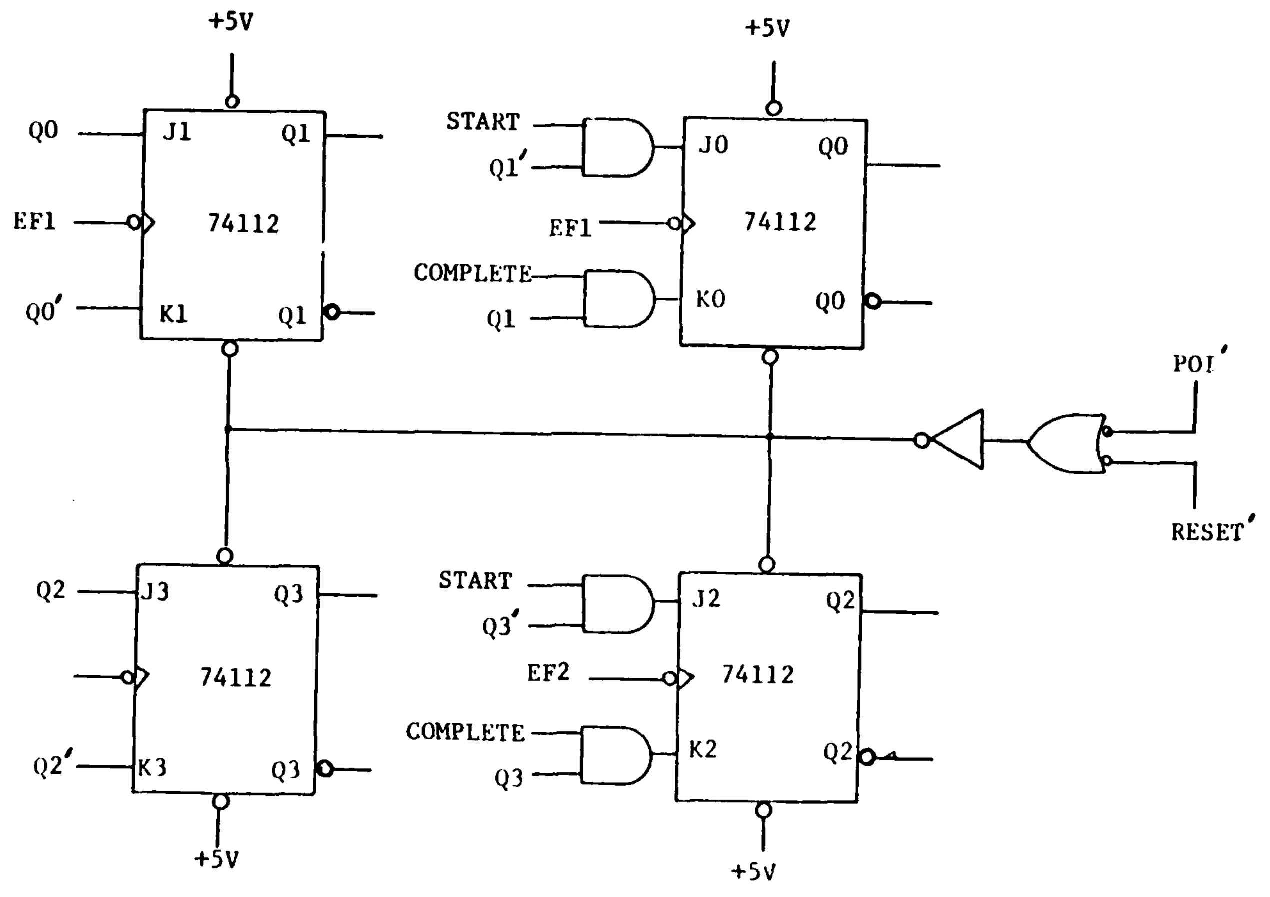


Figure 3.17 Implementation of the Image-Decoder Controller.

sent from the 11/70 to the vision system; detailed description about scanning control by the computer is given in Section 3.6. It is cleared by the negative pulse DS' used to strobe the UAR/T, by an external reset switch S2, or upon POI. The implementation is shown in Fig. 3.18.

The COMPLETE signal is set when both decoders arrive at state 11 of their own. It is cleared as decoder 1 goes back to state 00. The implementation is given in Fig. 3.19.

3.3.6.3 Image Coordinate Decoder

There is one image coordinate decoder associated with each camera. It receives intensity data of individual pixels from CCU, searches through an image frame and computes the X and Y coordinates of the first bright image whose intensity level is above an external threshold. The image coordinate system is shown in Fig. 3.20. There are 244 rows and 248 columns in a frame. Thus it requires 8 bits to represent X and Y coordinates respectively. The search is done on a row by row basis; i.e., the first row is scanned, followed by the second row, then the third row and so on. The block diagram of an image coordinate decoder is shown in Fig. 3.21. It consists of an X coordinate decoder, a Y coordinate decoder, and a comparator. Both decoders keep on counting until a pixel whose intensity level is higher than the threshold is encountered. At this point, the decoders stop counting and their values are preserved. A flag is set to indicate that a bright spot is detected. If no bright image is found at the end of a frame, the flag remains cleared and the content of the X decoder is 0 and that of the Y decoder is either 239 or 240. The image cordinates (X1,Y1) and (X2,Y2)

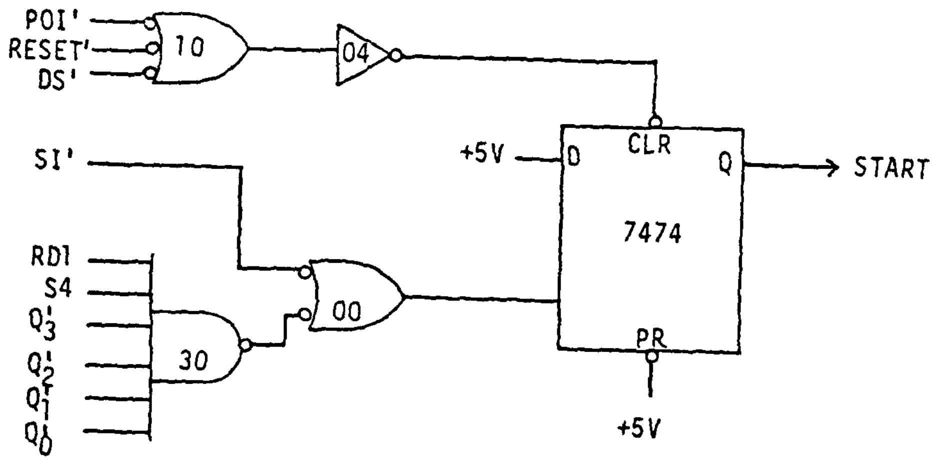


Figure 3.18 Implementation of the START Signal.

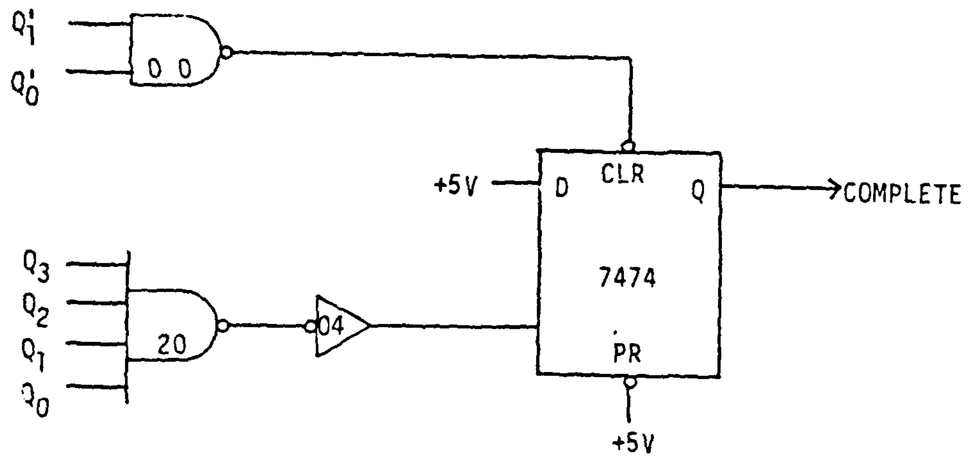


Figure 3.19 Implementation of the COMPLETE Signal.

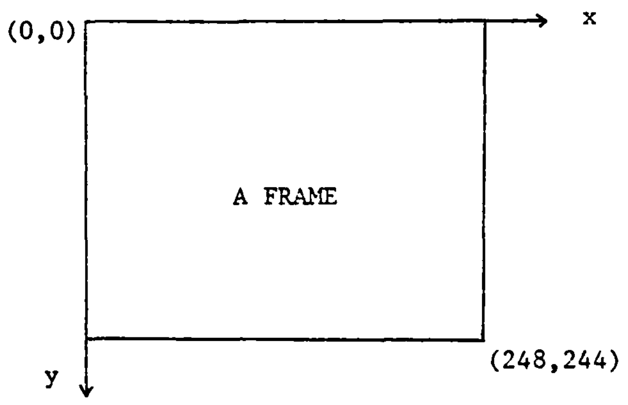


Figure 3.20 The Image Coordinate System.

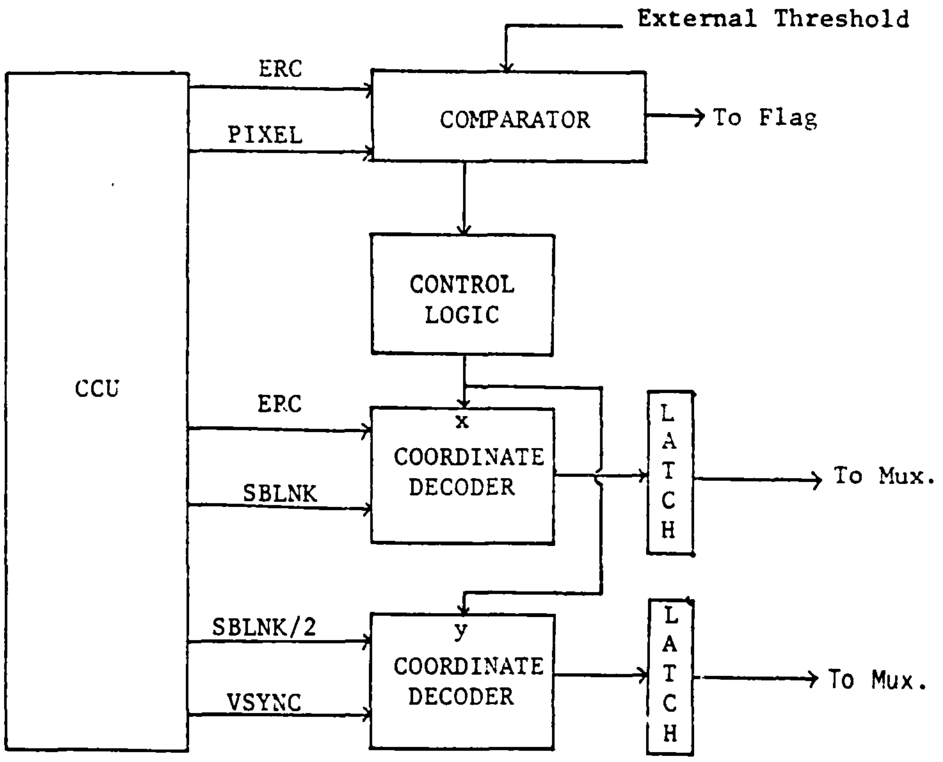


Figure 3.21 Block Diagram of an Image Decoder.

are then multiplexed and fed to the UAR/T. The control logic block is part of the interface controller described last section.

The 8-bit X coordinate decoder is cleared at the beginning of each row by the SBLNK signal, and is incremented by one for every pixel clock ERC. It was implemented by cascading two 74LS161 binary counters, as shown in Fig. 3.22. The contents of these counters are zero after POI or RESET. Furthermore, at a given instant the decoder is in one of the four states as explained earlier, and image detection occurs only in state 01. During the course of image detection within a frame, these counters are cleared at the very beginning of each row, which is initiated by SBLNK. On the other hand, if an image is detected the values of the counters should be preserved. Therefore SBLNK is AND'ed with Q1',Q0 and IM'. States Q1' and Q0 are presented to ensure that the decoder is indeed in the image detecting state. Signal IM' is a flag indicating if an image has been detected so far. If no image is encountered , IM' is high and the searching and clearing continues. If an image is detected, IM' becomes low and the clearing signal is disabled so the contents of the counters are stored. The ERC signal is fed to the CK pin of both counters so that it can count up the number of pixels in a row. The IM' and the state variables Q1' and Q0 are AND'ed and fed to the ENT (count enable) and ENP (carry look-ahead enable) pins of the lower counter so that it counts up only before an image is sensed. The ENT and ENP pins of the high nibble counter is connected to the CARRY pin of the low nibble one so that it can count up to 255.

The 8-bit Y coordinate decoder, shown in Fig. 3.23, differs from the X decoder in two aspects. First, it is cleared only once at the beginning of a searching frame, a condition realized by the AND'ed

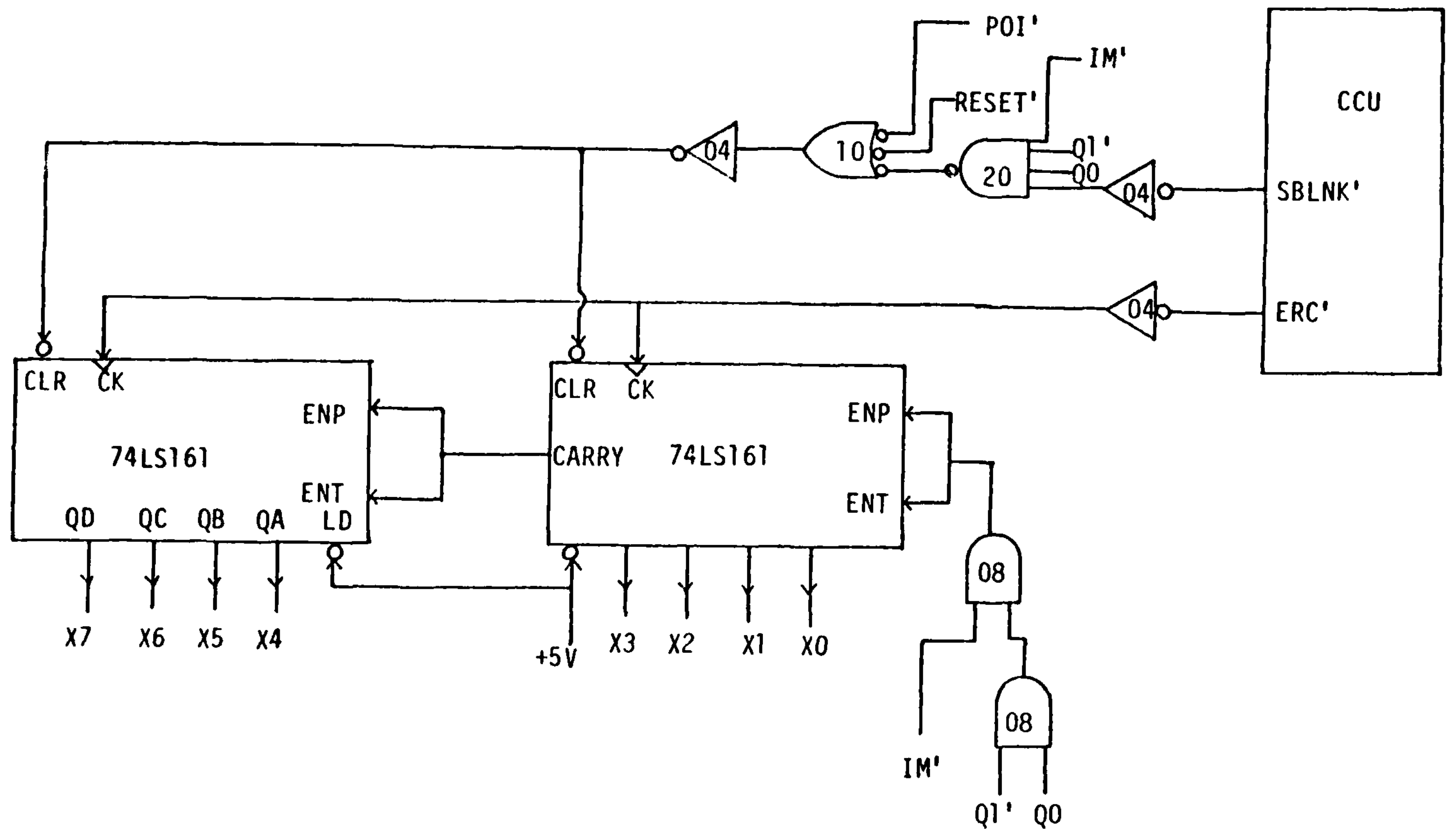


Figure 3.22 X-Coordinate Decoder.

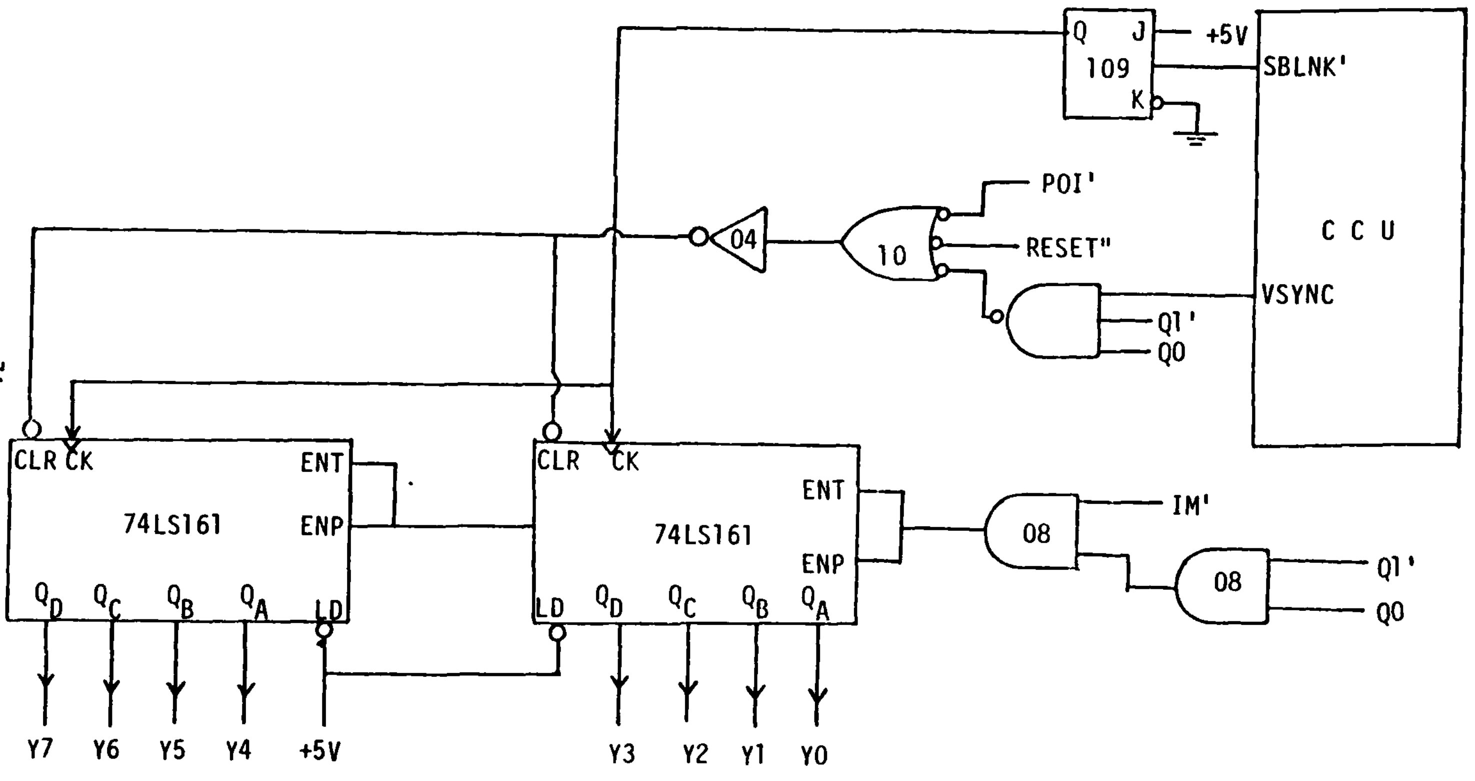


Figure 3.23 Y-Coordinate Decoder.

function of VSYNC, Q1' and Q0. Second, it is incremented by rows instead of pixels. Hence, the CK pin of the counters are connected to the SBLNK/2 signal. This signal is obtained by feeding SBLNK through a 74109 J-K flip-flop. It is necessary to divide the SBLNK signal by 2 because each row is scanned and displayed twice by CCU to accommodate the standard 525 line TV frame. The third bit of the Y counter is used to set the CROW signal for the purpose of skipping the first four rows because they do not contain useful information.

The 8-bit comparator is implemented by using two 7485's. The physical arrangement is shown in Fig. 3.24. Eight input pins, A0 to A7, are connected to a dip switch which provides the external threshold. Another eight pins, B0 to B7, are tied to the output of an 8-bit latch, 74273, which furnishes the intensity of the pixel currently being sensed. The A<B pin of the comparator is used to clear the no image flag IM'. The input of the latch is the 8-bit digital video signal coming from the CCU. It is very important to latch the intensity data at the right time so the CK pin is controlled by the AND'ed signal of ERC',Q1',Q0,IM' and CROW. The CROW signal is generated by the Y decoder to skip the first four rows as mentioned earlier.

3.3.6.4 UAR/T Controller

At the completion of an image detecting cycle, five bytes of data are generated. They are X1,Y1,X2,Y2, and a status byte. The organization of the status byte is shown in Fig. 3.25. A '1' in bit IM1 indicates that an image has been detected by decoder 1, otherwise, a '0' will be stored there. Bit IM2 carries similar information for decoder 2. A '1' in bit SCAN denotes that scanning operation under 11/70

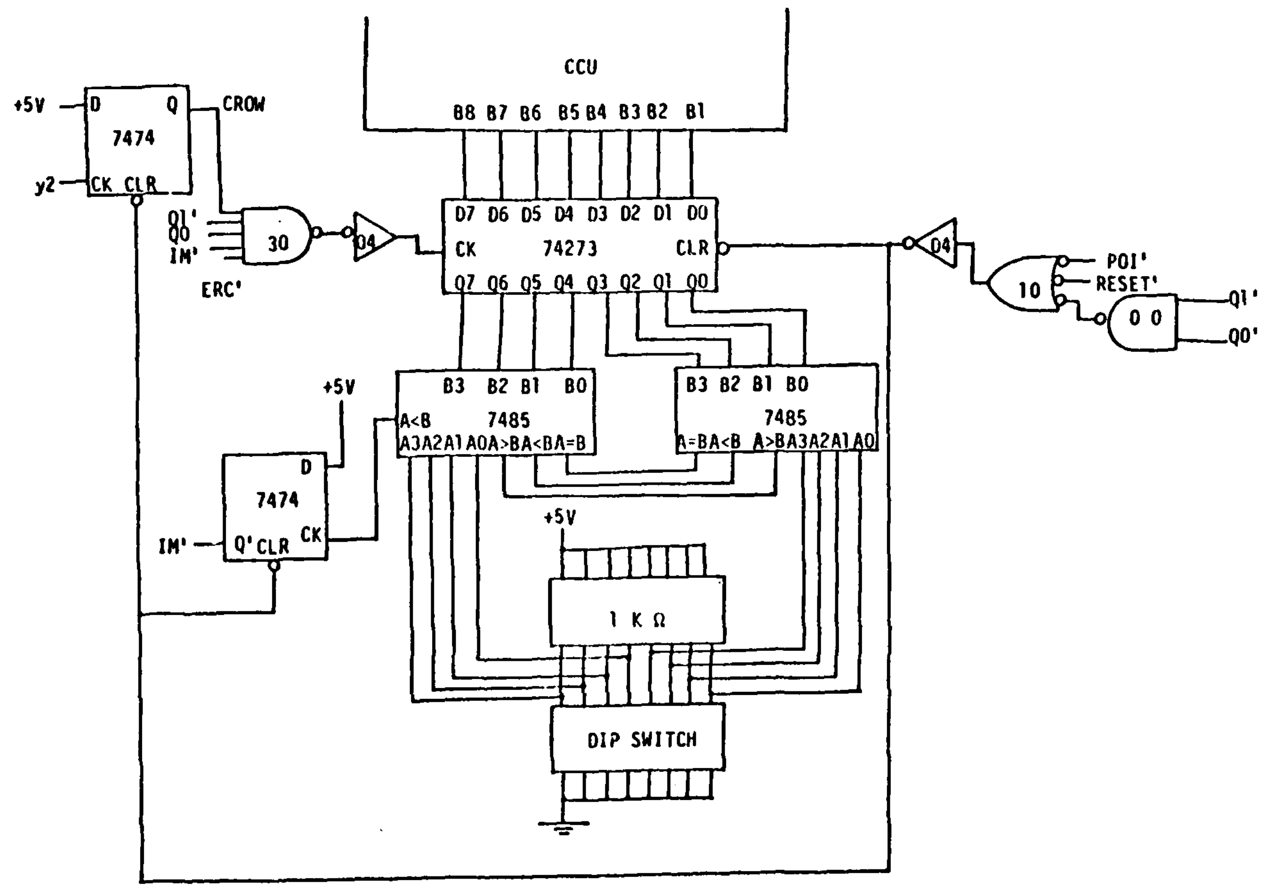


Figure 3.24 The Intensity Comparator.

control is in effect, while a '0' means that manual operation mode is adopted. A '1' in bit CONTINUE directs the 11/70 to keep on scanning and a '0' stop it. A '1' in bit READY informs the 11/70 that the image currently being sensed is a desired foothold and the triangulation routine should be called to compute the three dimensional coordinate.

These five bytes of data must be sent to the 11/70 computer for further processing. Technically, this can be done either in parallel or in serial. Parallel transmission is fast but more expensive because first, a DEC parallel interface card would be needed and second, a large cable is required due to the fact that every line connected to the 11/70 has to be optically isolated, which requires two wires for each line. Serial transmission is slower; nevertheless, the data rate of this application is low enough to be well handled serially. Therefore the serial scheme was adopted.

A UAR/T is used to facilitate data transmission and receiving. A sequential circuit, the UAR/T controller, is designed to oversee the function of the UAR/T and ensure that exactly 5 bytes are transmitted at the right time. The UAR/T chip used is a General Instrument Corporation AY-5-1012 product. It is a 40 pin LSI unit which accepts binary characters from either a terminal device or a computer and receives/transmits this character with appended control and error detecting bits. All characters contain a start bit, 5 to 8 data bits, one or two stop bits, and either odd/even parity or no parity. The baud rate, parity mode, and the number of stop bits are externally selectable. A diagram showing the pin configuration is given in Fig. 3.26.

The transmission of data to the 11/70 computer is regulated by a clocked sequential circuit, the state diagram of which is shown in Fig.

			IM2	CONTINUE	IM1	SCAN	READY

Figure 3.25 Organization of the STATUS Byte.

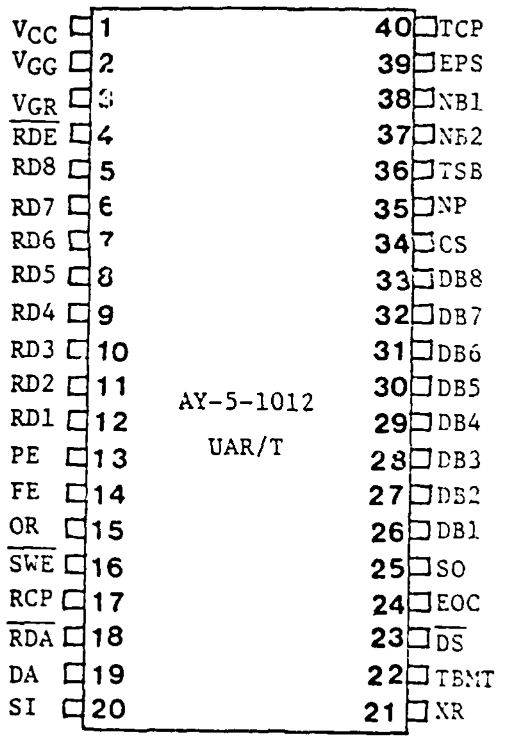


Figure 3.26 Pin Configuration of the UAR/T Chip.

3.27. Initially, it is idle at state A. The beginning of transmission is activated by the TRANSMIT signal, which is set to high in the same way as the COMPLETE signal (see Section 3.3.6.2), which signifies the end of the image detecting phase. When TRANSMIT becomes high, it enters state B. It stays at B for one clock cycle, clears all registers inside the UAR/T by pulling the XR (External Reset) pin high. Its activity in the following five states, C,D,E,F,and G are all similar. Upon entering each of these states, a negative pulse is generated and fed into the DS (Data Strobe) pin to have one byte of data loaded into the transmitter buffer of the UAR/T and then transmitted, while the controller waits there. When a byte is completely sent out, the TBMT (Transmitter Buffer Empty Flag) is set to indicate that the UAR/T is ready to accept another byte, and the controller moves into the next state. This action is repeated five times and exactly five bytes of data are transmitted one after one. The TRANSMIT signal is cleared during state H and the controller then goes back to state A. The flip-flop input equations can be easily derived. They are:

$$J2=Q1\bullet Q0'\bullet TBMT, \tag{3.10}$$

$$K2=Q1'\bullet Q0', \tag{3.11}$$

$$J1=Q2'\bullet Q0, \tag{3.12}$$

$$K1=Q2\bullet Q0\bullet TBMT, \tag{3.13}$$

$$J0=Q2\bullet Q1\bullet TBMT+Q2'\bullet Q1'\bullet TRANSMIT, \tag{3.14}$$

$$K0=Q2\bullet Q1'+TBMT\bullet Q2'\bullet Q1 \tag{3.15}$$

These equations were implemented by using three 74112 negative-edge triggered J-K flip-flops. The circuit diagram is shown in Fig. 3.28.

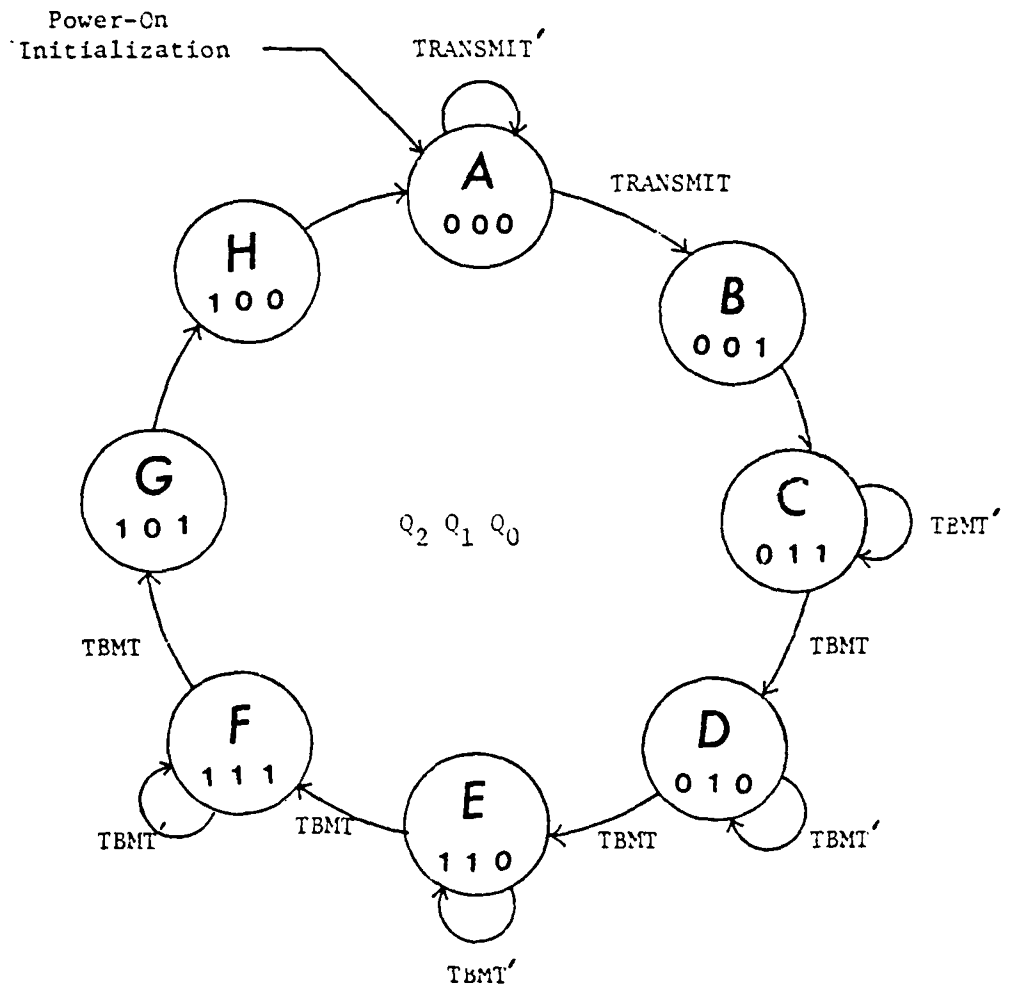


Figure 3.27 State Diagram of the Transmitter Controller.

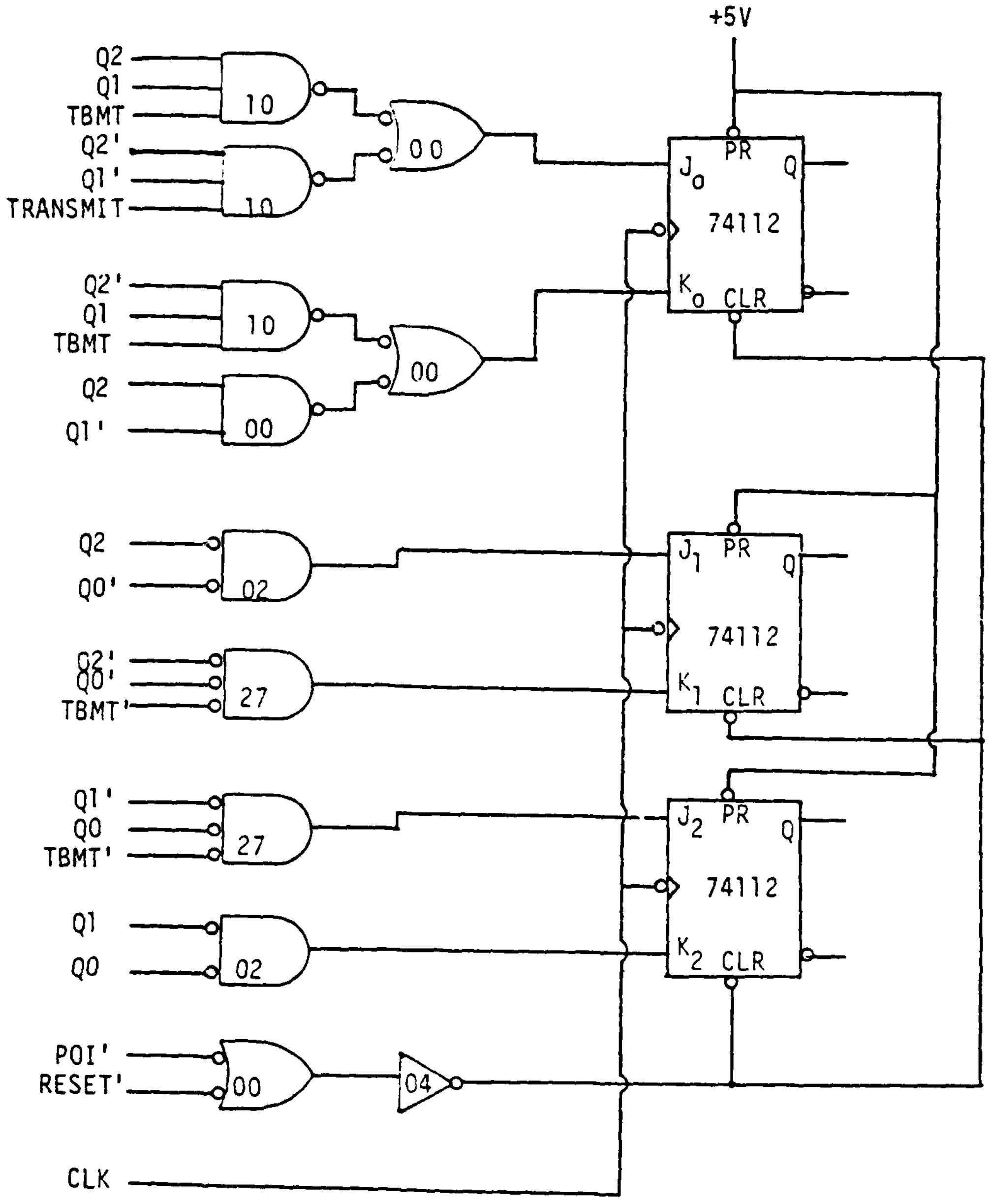


Figure 3.28 Implementation of the Transmitter Controller.

The clock signal which provides the timing base for the controller as well as the transmitter and receiver of the UAR/T is generated from a 555 timer, as shown in Fig. 3.29. The clock frequency can be adjusted in a range from 3.5 KHz to 100 KHz. The output of the 555 timer is fed to a 7414 Schmitt trigger to steepen the leading and trailing edges.

The negative data strobe required to load data into UAR/T is produced by a 74121 monostable multivibrator as shown in Fig. 3.30. The manufacturer requires that the data strobe be at least 250 nsec. in length. The resistor and capacitor chosen give a duration of 700 nsec. It is crucial that one and only one pulse be generated in each of the five states involved; therefore a 7474 flip-flop is dedicated to each state. Note that the data strobe is fed back to clear each 7474; otherwise, the output of 7430 would remain high once an involved state is travsersed and consequently only one pulse, instead of five, will be generated.

The five bytes of data are multiplexed and then fed to the UAR/T. It is arranged such that the ST (STatus byte) is loaded first during state C, followed by X1, Y1, X2, and Y2. This portion of the circuit is shown in Fig. 3.31.

The part of controller which guards the receiving function is much simpler. There are only two states involved, and its state diagram and implementation are shown in Fig. 3.32. It remains idle in state 0 until a byte has been received and assembled. This event is signaled by UAR/T's pulling the DA (Data Available) line high. When this occurs, it moves into state 1 where the received byte is latched onto an 8-bit register (74273). After one clock cycle, it resets the data available flag and returns to state 0. Although one byte of data is received from

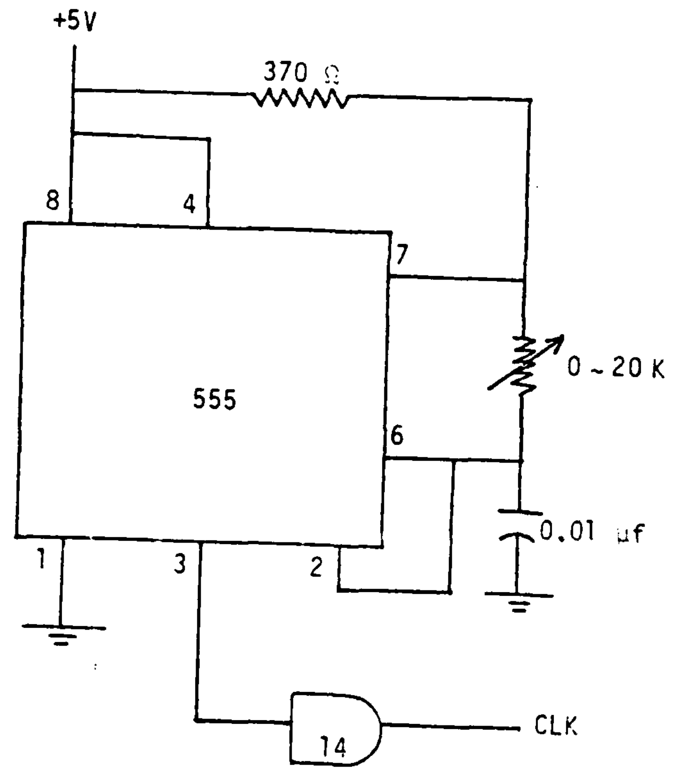


Figure 3.29 Clock Source Using a 555 Timer.

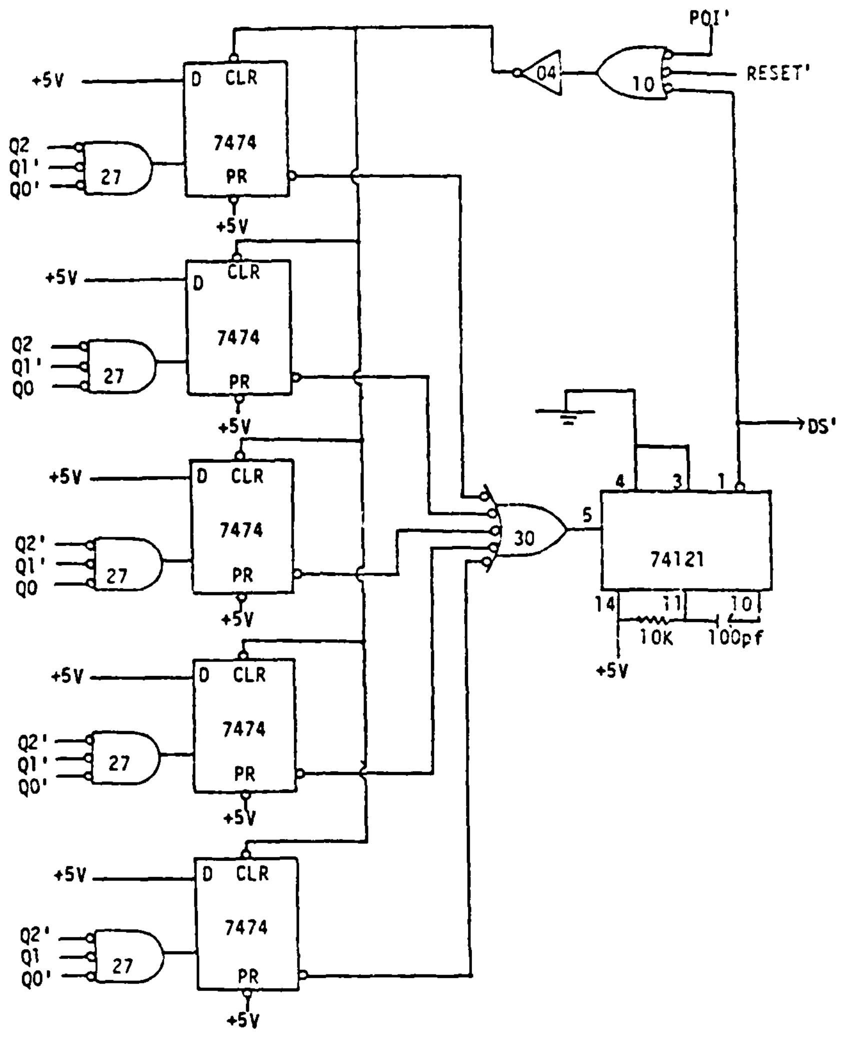


Figure 3.30 The Strobe Generator.

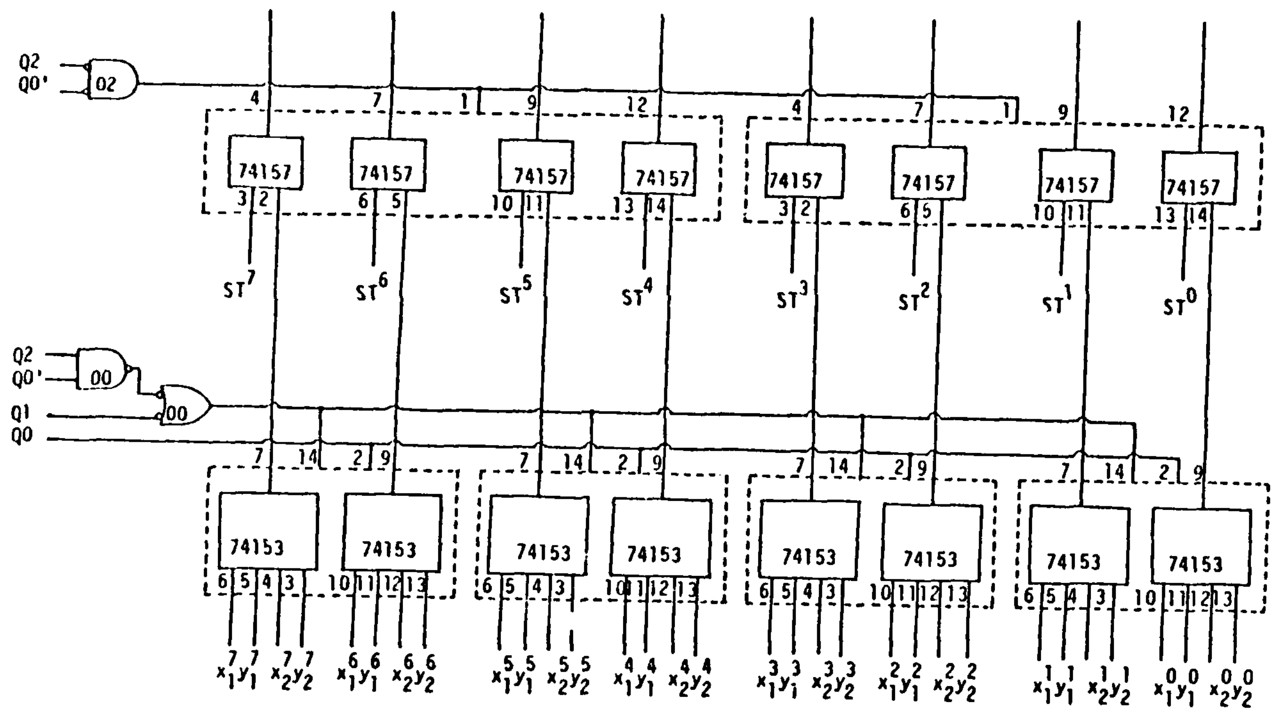
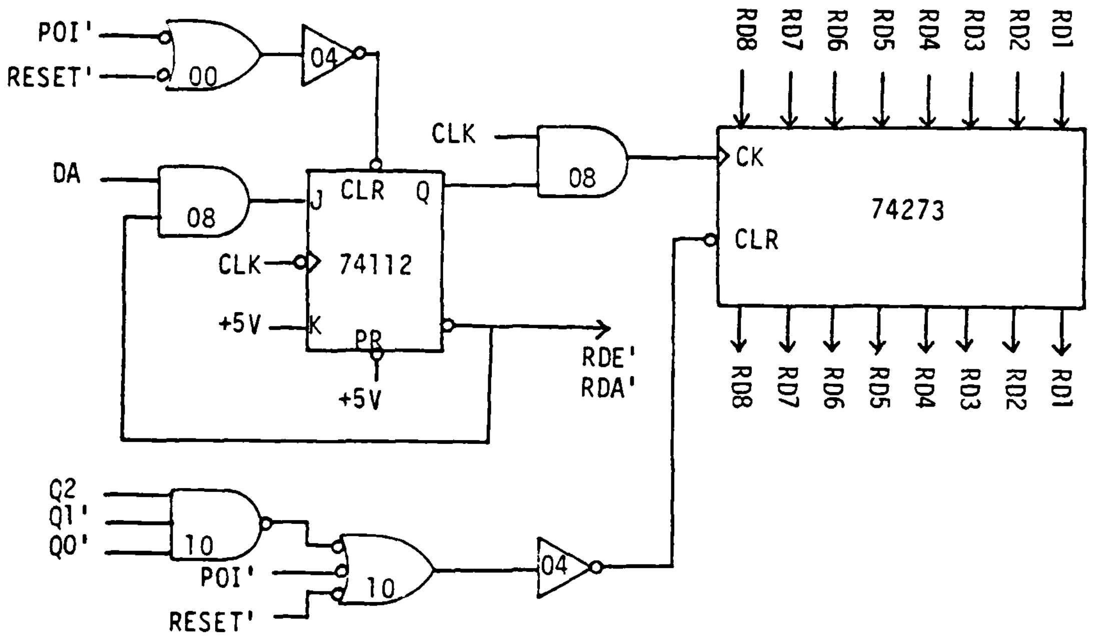
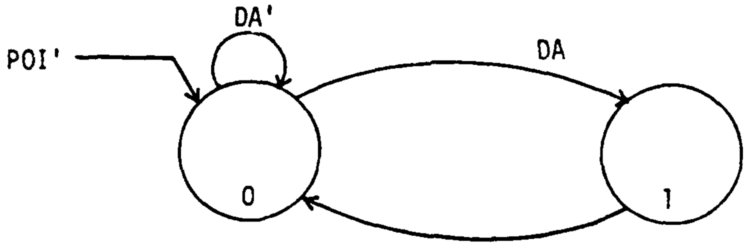


Figure 3.31 The Multiplexer.



(a)



(b)

Figure 3.32 The Receiver Controller.
(a) Implementation; (b) State Diagram.

the 11/70 each time, only two bits carry information. A '0' to '1' transition in RD1 triggers an image detecting operation (recall Fig. 3.18), and a '1' in RD2 clears the READY flag in the status byte. The READY bit is set by the operator through a remote-controlled switch to indicate that a foothold has been selected.

The POI (Power-On Initialization) signal was obtained by feeding a +5V source through a 1K resistor and a 30 μF capacitor, as shown in Fig. 3.33. The external RESET signal was implemented using a debounced toggle switch as shown in Fig. 3.34.

3.3.6.5 Optical Isolation

To protect the 11/70 computer from damages resulting from possible abnormal function of an external circuit, the serial communication channel is optically isolated as shown in Fig. 3.35. Data from the SO (Serial Output) pin of the UAR/T is fed through a MC1488P line driver to meet the RS-232 standard, which demands voltage levels of $\pm$ 15V instead of the $\pm$ 5V TTL level. The output of the line driver is connected to a HP2630 optocoupler. In a short circuit situation, the diode in the input side of HP2630 will be burnt out but the other side of circuit will not be affected because there is no physical connection between them. Under normal conditions, the input current passing through the diode will activate the other side of circuit. Since the output of a HP2630 is TTL compatible, another line driver is needed to convert it into RS-232 level. Data sent out from the 11/70 is optically isolated as well, and is received at the SI (Serial Input) pin.

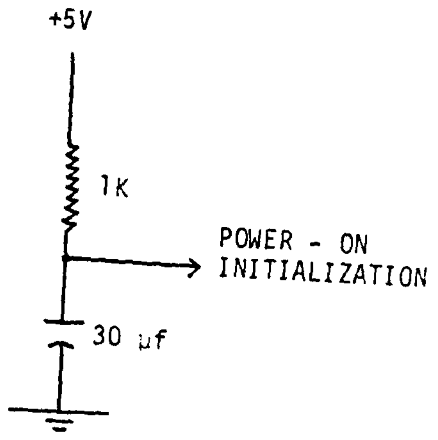


Figure 3.33 The Power-On Initialization.

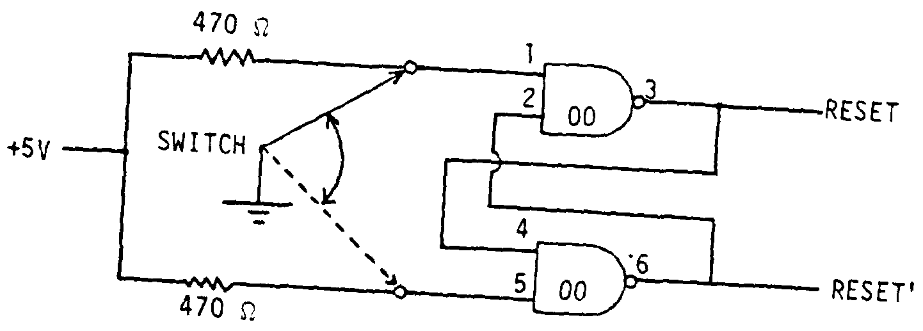


Figure 3.34 The External RESET.

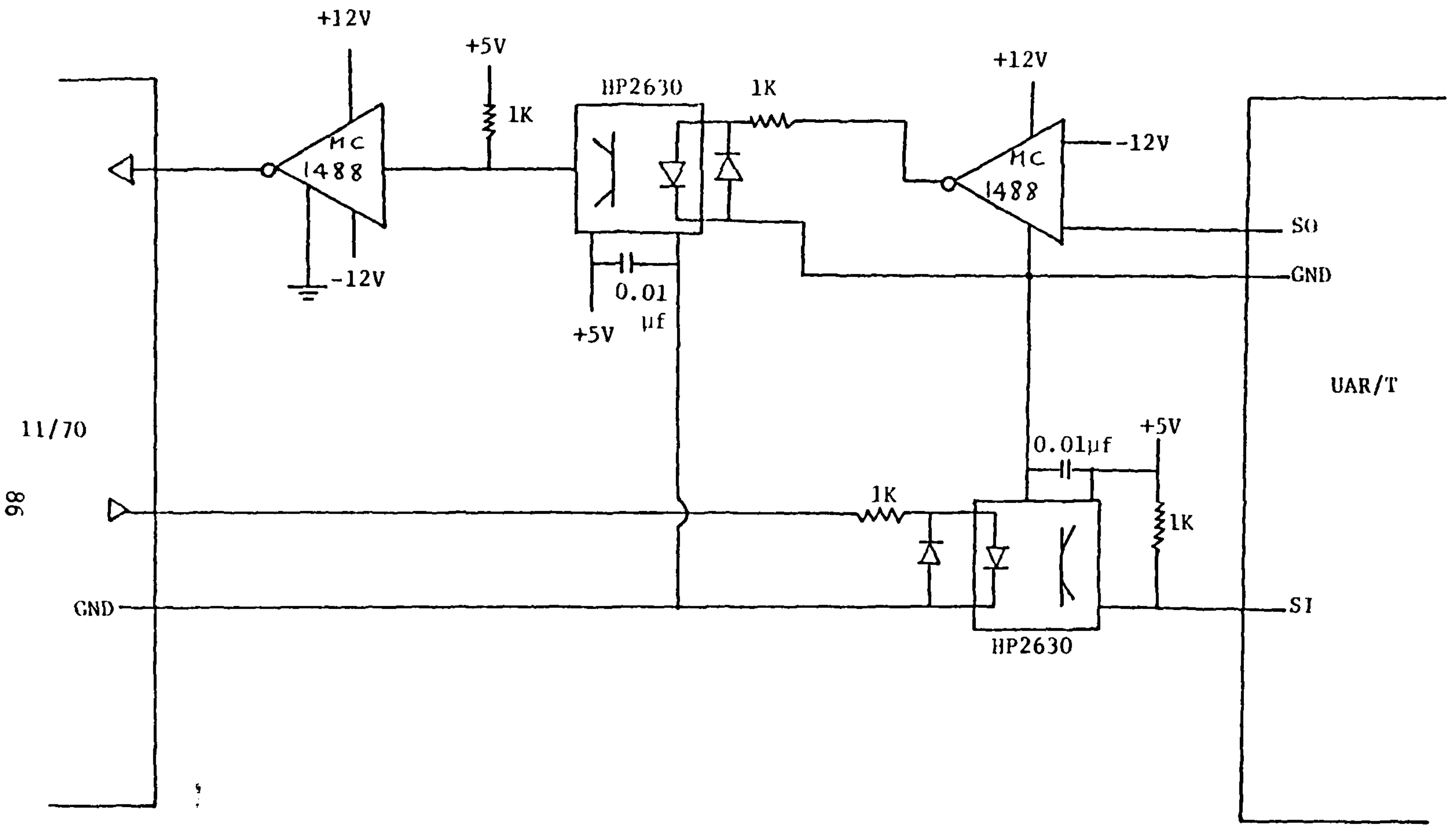


Figure 3.35 Optical Isolation Circuitry.

3.3.6.6 The Interface Circuit Board

The interface circuit was constructed on an AUGAT UL-1 circuit board. Its layout is shown in Fig. 3.36, and a picture of the the physical board is given in Fig. 3.37. The signals from CCU1 are connected to connector J1 of group A through flat cable C1. Similarly, signals from CCU2 are fed to connector J1 of group B by using C2. Connector J1 of group C is tied to the TT3 terminal connector through C3,and the ±12V from the power supply unit are connected to J1 of group D. The image coordinates X1 and Y1 are made available at J2 of group A, and so are X2 and Y2 at J2 of group B. Port J1 of group E is connected to a remote, push-button switch which is used by an operator to set the READY flag of the status byte when laser beam has been focused on a desired spot.

There are five switches on board. Switch S1 is used to trigger the image detecting operation in the manual control mode. A positive transition of S1 would cause the circuit to search for an image once. Switch S2 is the external reset switch. When S2 is high, both image decoders and the interface controller are reset to their initial ststes. Switch S4 indicates whether it is a manual or a scanning operation. With S4 down, the operation is controlled manually through S1. With S4 and S5 both up, the circuit would continue scanning under computer control. Pull S5 down would terminate the scanning. Switch S3 is not used.

There are four dip switches denoted as U10, U23, U30 and U43 respectively. Among them, U10 and U23 are associated with camera 1 and the other two with camera 2. Switch U10 is used to control G1, G2, IIG and I/244SEQ pins of camera 1 so that it is operated in a sequential

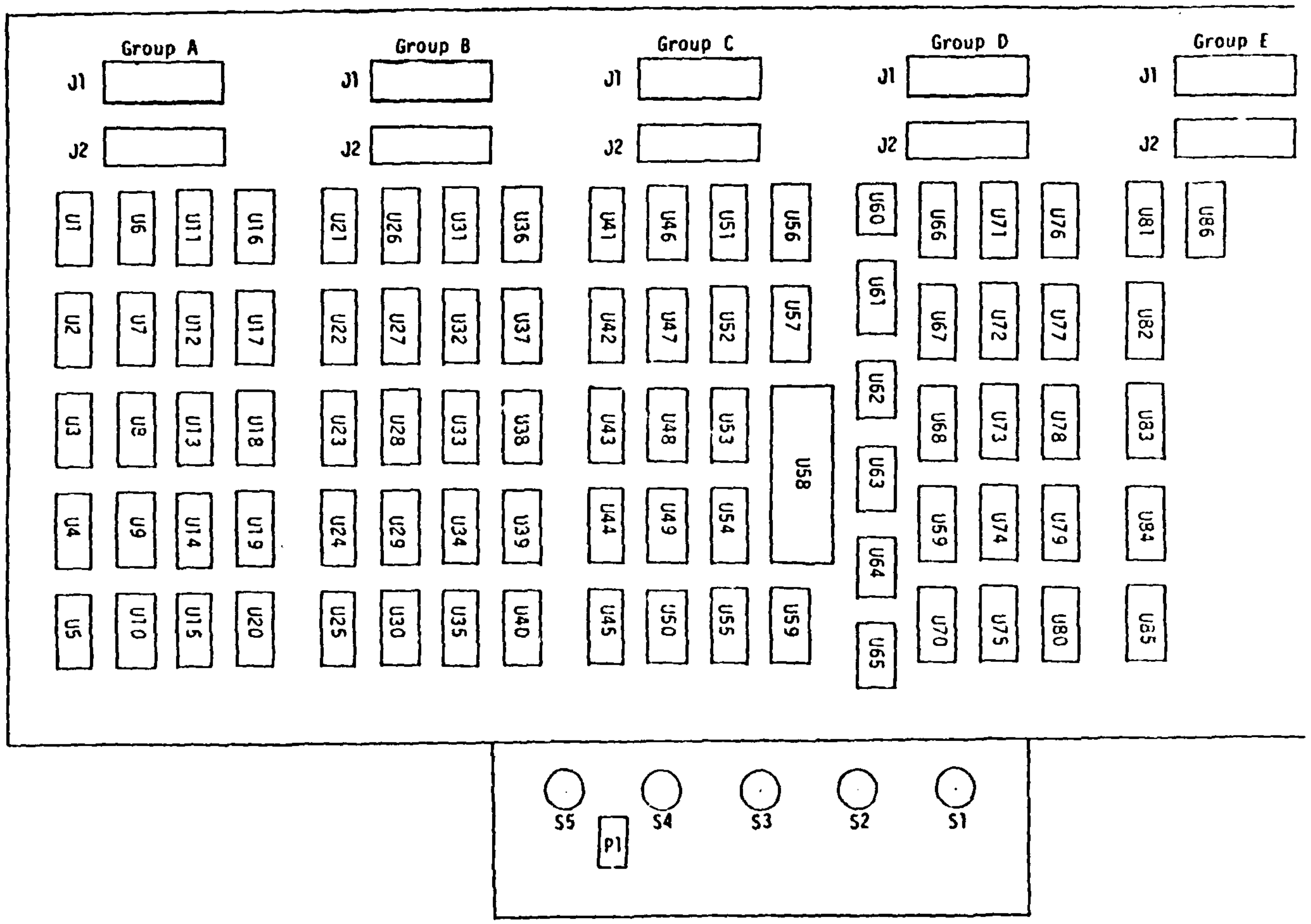


Figure 3.36 Layout of the Interface Board.

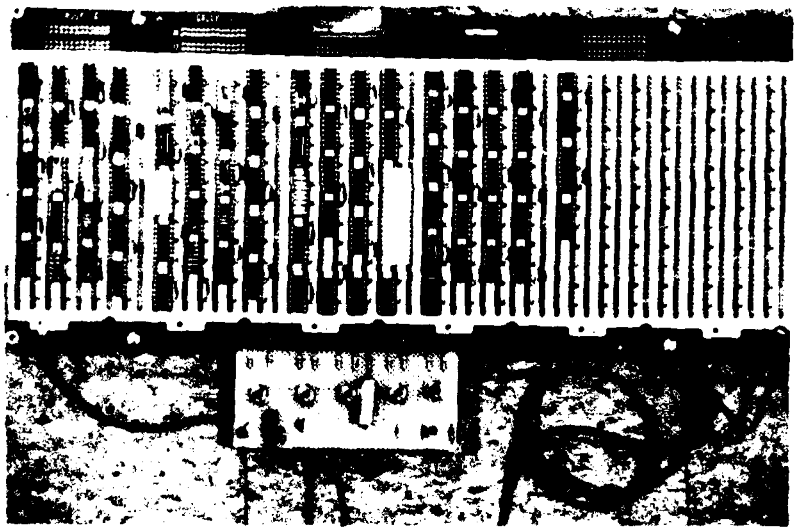


Figure 3.37 Picture of the Interface Circuit Board.

mode with injection inhibit disabled. Pins G1 and G2 are both tied low so that all the clock and digital output signals are always available. Switch U23 is the external threshold used to detect a bright image. The functions of U30 and U43 are exactly the same as those of U10 and U23.

The variable resistor pot P1 is used to adjust the clock frequency of the UAR/T to accommodate a chosen baud rate.

3.4 Human Interaction

The responsibility of the human operator is to select proper footholds for the Hexapod. To accomplish this function reliably, a communication channel is needed so that the operator can tell the vision system, which is searching for an image point constantly, whether this is the image of a desired foothold. One way to handle this problem is to impose a "hand-shaking" policy, which requires the operator to set a READY flag for a selected point and has the vision system clear it after the image is processed. Physically, a remote-controlled, push-button switch connected to the interface board through a long cable is held by the operator. When the switch is pushed, its output signal is first debounced then fed to a flip-flop (74112) whose output is the READY bit of the status byte. This part of circuitry was given in Fig. 3.38. The 11/70 computer, after computing the coordinates of that foothold, immediately sends out one byte of data and uses the second bit (RD2) to clear the READY flag.

3.5 Operational and Testing Procedures of The Vision System

To insure that the vision system functions properly, the procedures listed below should be followed.

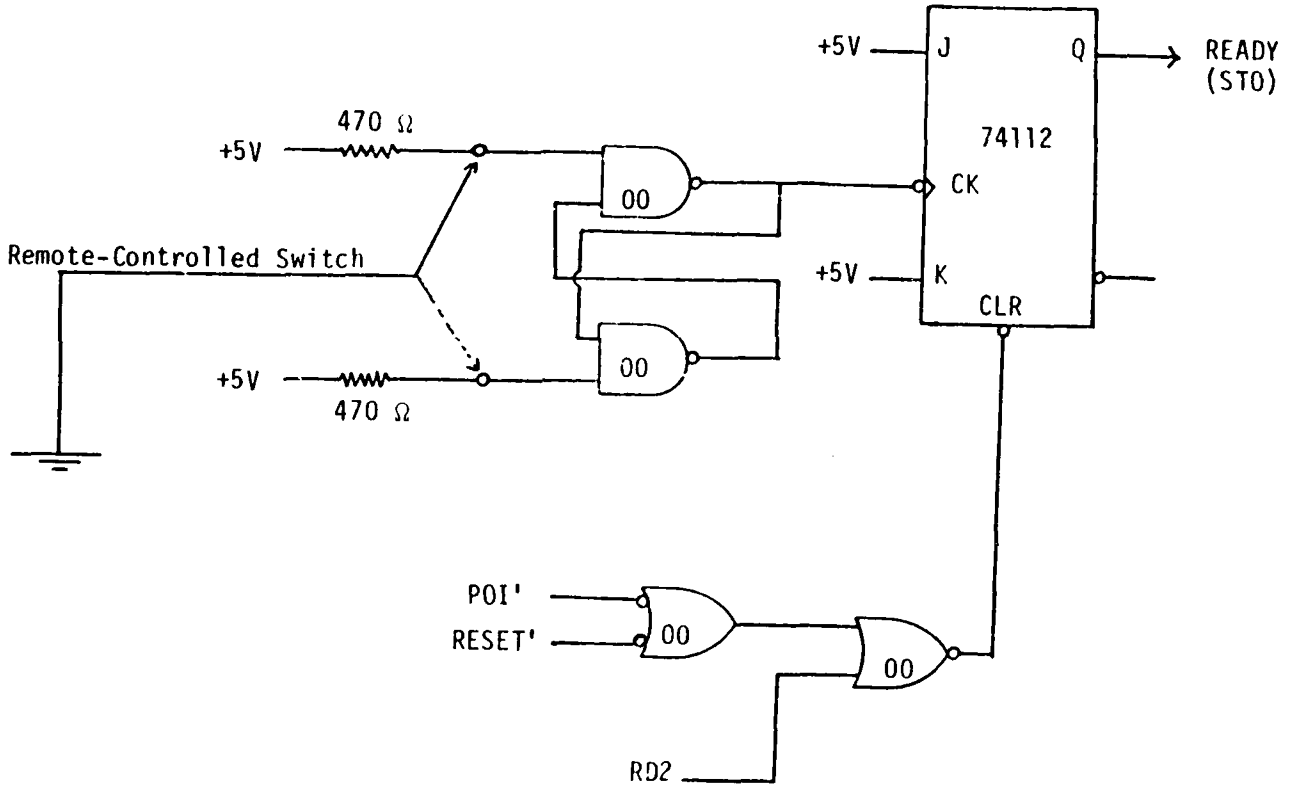


Figure 3.38 READY Flag and the Remote-Controlled Switch.

(1) Select a baud rate and adjust the clock frequency accordingly. A selectable range of baud rate from 0 to 9600 is available [50]. For this application, a baud rate higher than 600 is sufficient. After choosing a particular rate, one should adjust pot P1 on the interface board and measure the clock frequency by an oscilloscope or a frequency meter. Note that the tolerable clock skew of DH-11 is 4.8%, thus the frequency of the transmitter and receiver of the UAR/T should be accurate to that extent. The system has been operated at a baud rate of 1200 with highly stable results.

(2) Set up equipment. One end of cable C1 is connected to the interface connector of CCU1 and the other end to the J1 connector of group A on the interface board. Cable C2 should be connected to the similar location of CCU2 and J1 of group B. Cable C3 ought to be connected between J1 of group C and the TT3 terminal connector, and the power lines at J1 of group D are tied to the $\pm 12V$ of the power supply unit. The remote-controlled switch must be connected to the J1 port of group E.

(3) Set the apertures of both lenses to be minimum (f/16). With this aperture, a proper threshold for image intensity is 12, which is high enough to supress internal noise and low enough to be overriden by the desired image.

(4) Enter the following commands at the system console:

```
>set /speed=tt3:1200:1200 (assume that baud rate is 1200),
>set /slave=tt3:
>set /fdx=tt3:
```

The speed command notifies the 11/70 computer that the vision terminal which was assigned as TT3 is operating at a baud rate of 1200. The slave command causes the vision terminal to be treated as a slave while tne 11/70 is the master. A slave terminal will not send any message toward tne computer unless it is asked by the master to do so. With this arrangement, any unsolicited data generated by noise will be labelled as such and the computer is able to differentiate image data provided by the vision terminal from noise. The fdx command establishes the communication link between TT3 and the 11/70 as a full duplex channel.

Enter the following command from the terminal that one logs in:

```
>ASN TT3:=OT:
```

This command informs the computer that TT3 is the input as well as the output terminal associated with the I/O requests appearing in the program.

(6) Set up the laser head and laser power supply.

(7) Turn on the power supply and both CCU's.

(8) Pull switch S2 up and down to reset the vision system. Set switches S4 and S5 in the 'up' position to put it in the scanning mode.

(8) To test if the vision system functions normally, run program VID. If unsolicited data exits, the program will abort itself and VID has to be run again. Otherwise, the message "SPECIFY FOOTHOLD" is displayed on the screen to prompt the operator for a foothold. The laser unit should be turned on and the beam be aimed to a point which is within the field of view of both cameras; then the remote switch is pushed. If the image is sensed by both cameras, the X, Y, and Z coordinates will be displayed on screen, or else the character string "NO IMAGE, TRY AGAIN" is shown and the operator has to adjust the laser head and then activates the remote switch again.

3.6 Scanning Control Of The Vision System

The software which deals with the interaction between the 11/70 and the interface circuit and enables the computer to control the scanning of the vision system deserves special explanation because (1) it involves activities embedded at a lower level and requires assembly language programming, which is not as easy to comprehend as codes written in PASCAL, and (2) it utilizes many of the RSX-11 system directives. Although detailed documentation of them are provided by [51], a brief description of each directive whenever it is encountered would make it easier to follow. Therefore, the detail of program VID, which manages the interaction of the 11/70 and the vision system to facilitate scanning control, is examined here. Its block diagram is shown in Fig. 3.39.

First of all, the vision terminal has to be logically connected to the 11/70 system so that the operating system can keep track on its

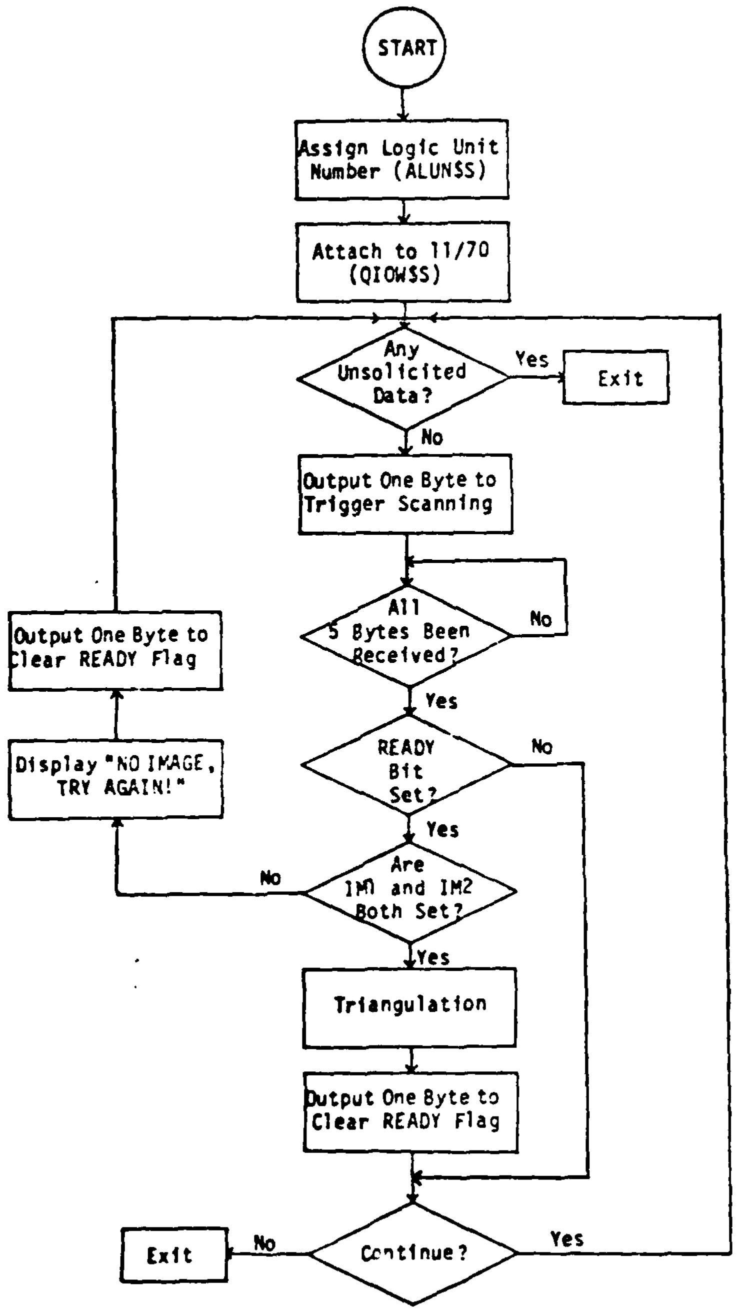

START
Assign Logic Unit Number (ALUN$S)
Attach to 11/70 (QIOW$S)
Any Unsolicited Data?
Yes
Exit
No
Output One Byte to Trigger Scanning
All 5 Bytes Been Received?
No
Yes
READY Bit Set?
No
Yes
Are IM1 and IM2 Both Set?
No
Yes
Display "NO IMAGE, TRY AGAIN!"
Output One Byte to Clear READY Flag
Triangulation
Output One Byte to Clear READY Flag
Continue?
No
Exit
Yes

Figure 3.39 Flowchart of Program VID.

status. This is done through the following statements:

```
LUN = 9.
ALUN$S  #LUN,#"OT,#0
QIOW$S  #IO.ATT,#LUN
```

The variable LUN represents the particular logical unit number that the vision terminal is assigned. The execution of directive ALUN$S instructs the system to assign a physical device unit to a logical unit number. Thus, the physical device whose name is "OT" is assigned a logical unit number of LUN, which has a value of 9. Recall that the vision terminal is TT3, hence if the command [ASN TT3:=OT:] is issued from a VT100 terminal console, the vision terminal will has a logical unit number of 9. The next instruction, QIOW$S, instructs the system to place an I/O request for the device with the logical unit number LUN and wait until the request is fulfilled. The subcommand #IO.ATT specifies that this particular I/O task is to attach this device to the system. After it is executed, the communication link between the 11/70 and the vision terminal is logically connected and completely established.

The next step is to check whether unsolicited data exists. Every I/O device is allocated an input and an output buffer by the operating system. (The directions of input and output are referred to with respect to the 11/70.) The input buffer of a slave device should be empty if no input request has been issued by the computer. The only occasion that the 11/70 would ask the vision system to send in data is during foothold acquisition. Thus, before it initiates such an event, any data in the input buffer is interpreted as unsolicited data. In other words, the number of bytes of data in the input buffer must be zero prior to an input request. This quantity may be accessed by using the following directive:

```
    QIOW$S  #SF.GMC,#LUN,#EF1,,,,<#STAD,#2>
```

Again this is an I/O request instruction because it involves the vision terminal. The subcommand #SF.GMC directs the system to return the information of device #LUN. Particularly, if the variable STAD is initialized to be TC.TBF by using the BYTE directive, the returned value gives the number of bytes of data in its input buffer. If this value is not zero, then unsolicited data exists and the program will abort itself.

To initiate an image detecting event, the computer sends out one byte of data to the vision terminal and uses the least significant bit to trigger the interface circuit. This output operation is achieved by the following code:

```
    QIOW$S  #IO.WAL,#LUN,#EF1,,#IOST,,<#WRBUF,#1>
```

The subcommand #IO.WAL orders the system to write (or output) data to device #LUN. Moreover, parameter <#WRBUF,#1> indicates that only one byte is to be sent out and that this byte of data is located at WRBUF. The process of the program will be halted until the output event is completed as signaled by the event flag EF1. The status of this request is made accessible at location IOST.

While the vision system is searching for an image, the computer prepares for the arrival of 5 bytes of data as the result of image searching by declaring another I/O request:

```
QIOW$S  #IO.RAL,#LUN,#EF1,,#IOST,,<#RDBUF,#5>
```

This command instructs the system to read 5 bytes of data from device #LUN and stores them at location RDBUF. The subcommand #IO.RAL simply indicates that this is an input request. The meanings of EF1 and IOST are the same as described earlier.

Among the 5 bytes of received data, the status byte is investigated first. Above all, the READY bit is checked to see if this is the image of a specified foothold. If not, nothing will be done except examining the CONTINUE flag to decide whether the scanning operation is to be continued. In case that the READY bit is set, which implies this image is of interest, IM1 and IM2 are checked to determine whether the image is detected by both cameras. If either one or both miss it, a message is displayed on a screen to prompt the operator to give another try. In addition, the 11/70 will send out one byte of data and use the second bit (RD2) to clear the READY flag. If both IM1 and IM2 are set, triangulation is performed to compute the three-dimensional coordinates of the detected image. This subject is discussed in depth in the next chapter.

3.7 Summary

In this chapter, problems encountered in realizing FTL locomotion were identified and possible solutions were explored. After comparisons of different alternatives, the approach which employs a binocular vision system was deemed more suitable than others and thereby was adopted. To control the front feet of the Hexapod, an operator points a laser beam at terrain surface points, which are then perceived by the vision system as the footholds.

The problem of selecting camera mounting locations was analyzed in consideration of the field of view and mechanical stability. It was concluded that 6.5 mm wide-angle lenses were needed in order to satisfy the requirements.

Both narrow-band and broad-band optical filters have been tested to study their noise rejection capability. The former indeed blocked out most of the ambient noise but also severely attenuated the returned laser signal, while the latter demonstrated an inferior noise rejection capability. It was found that, inside a laboratory and under normal lighting condition, the most efficient way to reject ambient light while maintaining sufficient intensity for reflected laser beam is to reduce the aperture of the lens to the minimum value of f/16 with no filters on it. This may be due to the nonlinearity of the internal electronic circuits of the camera heads. Extermely strong light sources should be avoided since they will not be totally rejected even with a minimum aperture.

A special-purpose hardware circuit was designed and constructed to compute the image coordinates of an object point in both camera frames and to send the data to the 11/70 computer. Details of the design and implementation were presented.

The vision system is treated by the 11/70 as a regular peripheral terminal. Their communication is based on a master-and-slave, handshaking scheme. The software which enables the 11/70 to control the scanning operation of the vision system has also been described.

Chapter 4

CALIBRATION OF THE VISION SYSTEM

4.1 Introduction

The determination of the spatial coordinates (X,Y,Z) of a foothold takes two steps. First, the image coordinates in both camera frames are obtained through the interface circuit. Then, a projection ray is identified for the image point on each frame and the intersection of these two rays yields the location of the foothold, a process which is known as triangulation. The accuracy of the result depends on how precise these projection rays are defined. This requires a proper camera model, an accurate measurement of the internal geometric parameters of the camera, and compensation to correct for lens distortion. Furthermore, the coordinates computed have to be transformed from the camera CS (Coordinate System) to the Hexapod body CS, which calls for the identification of the relation between the camera and the body CS's. These subjects are discussed in this chapter. A pinhole camera model is introduced in Section 4.2. Based on this model, the mapping between a 3-D object point and its corresponding 2-D image point is formulated. In Section 4.3, procedures for camera calibration are described. In Section 4.4, distortion caused by a wide-angle lens is characterized by a polynomial and compensated accordingly. In Section 4.5, the triangulation technique is described. The arrangement of camera installation by using two ball-socket heads and its effect on the common field of view are presented in Section 4.6. The geometrical relationships between cameras and the Hexapod body are identified in Section 4.7, with cameras being attached to the mounting frame by using fixtures instead of ball-socket heads.

4.2 The Camera Model

The purpose of an analytic camera model is to relate 3-D coordinates of points in object space to the 2-D locations of their images in a digitized picture frame. An adequate and widely used model is the so-called pinhole model in which the camera is represented by a pinhole lens together with an image plane lying at distance of F behind the lens, where F is the focal length of the lens. The model adopted here is very similar to a pinhole model except that the image plane is placed in front of the lens center to avoid the inconvenience of inverted images. Fig. 4.1 shows such a camera model with a coordinate system assigned to it. The origin of this camera CS is chosen at the upper-left corner so that the image coordinates agree with those that are generated by the image coordinate decoders. Note that the Z coordinate is 0 for every point on the image plane. The coordinates of the lens center in this coordinate system are denoted as (Xc,Yc,-F).

Based on this model, the relationship between an object point and its image can be readily established. Given any point (X,Y,Z) in the object space, its image (Xp,Yp) can be uniquely determined; and knowing an image (Xp,Yp), one can identify the corresponding projection ray. Referring to Fig. 4.1 and assuming that the 3-D coordinates of object

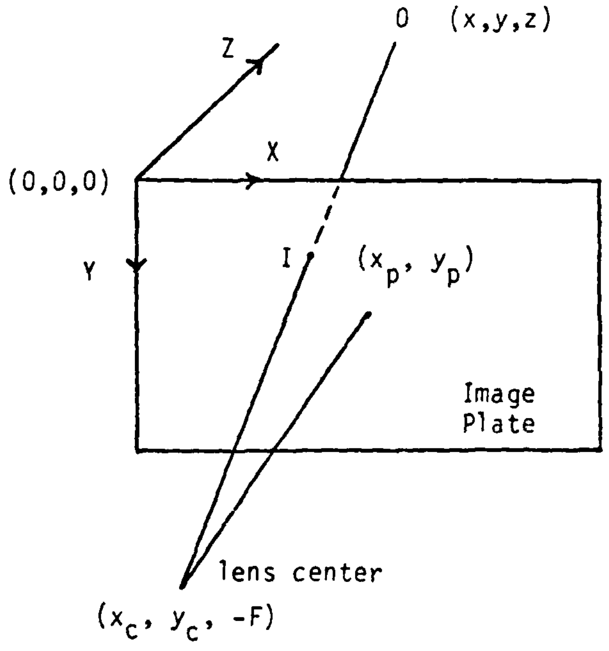


Figure 4.1 The Camera Coordinate System.

point O are (Xo,Yo,Zo), the straight line passing through O and the lens center C can be characterized by the following equation:

$$\frac{Xo-X}{X-Xc} = \frac{Yo-Y}{Y-Yc} = \frac{Zo-Z}{Z+F} \tag{4.1}$$

In this equation, (X,Y,Z) are the coordinates of an arbitrary point lying on the segment CO. Obviously, the image point (Xp,Yp,Zp) is one of such points. Thus

$$\frac{Xo-Xp}{Xp-Xc} = \frac{Yo-Yp}{Yp-Yc} = \frac{Zo-Zp}{Zp+F} \tag{4.2}$$

Recall that the Z coordinate of any point on the image plane is 0, i.e.;

$$Zp=0 \tag{4.3}$$

Substituting Eq. (4.3) into (4.2) results in

$$\frac{Xo-Xp}{Xp-Xc} = \frac{Zo}{F} = \frac{Yo-Yp}{Yp-Yc} \tag{4.4}$$

Therefore

$$X_p = \frac{(Z_O/F) \cdot X_C+X_O}{1+(Zo/F)} \tag{4.5}$$

$$Yp = \frac{(Zo/F) \cdot Yc+Yo}{1+(Zo/F)} \tag{4.6}$$

Since the picture is digitized, the image coordinates IX and IY are integers with

$$IX = ROD(Xp/Dx) \tag{4.7}$$

$$IY = ROD(Yp/Dy) \tag{4.8}$$

Here ROD is the round-off function, Dx is the length of a pixel in the X direction and Dy is the length in the Y direction.

On the other hand, if an image (IX,IY) is detected and it is desired to find out the projection ray, one may proceed as follows. Assuming that the image is located at the center of the pixel (IX,IY), then its coordinates are

$$Xp = Dx \cdot (IX-1/2) \tag{4.9}$$

$$Yp = Dy \cdot (IY-1/2) \tag{4.10}$$

$$Zp = 0 \tag{4.11}$$

The projection ray is expressed as follows:

$$\frac{X-Xp}{Xp-Xc} = \frac{Y-Yp}{Yp-Yc} = \frac{Z}{F} = K \tag{4.12}$$

which can be written into a set of parametric equations

$$X = Xp+K \cdot (Xp-Xc) \tag{4.13}$$

$$Y = Yp+K \cdot (Yp-Yc) \tag{4.14}$$

$$Z = K \cdot F \tag{4.15}$$

with K as the parameter.

4.3 Camera Calibration

To accurately locate a point in space by using vision, the relation between a point on the image plane and the projection ray associated with this point should be known precisely. This relation was established in the last section by using a pinhole model with a front image plane. During the derivation, the coordinates of the lens center (Xc,Yc,-F) were assumed to be known. From Eqs. (4.5), (4.6), (4.13), (4.14) and (4.15) it is clear the mappings between images and object points are functions of these parameters. Hence, it is important to precisely determine the location of the lens center as represented in the camera CS. These internal geometric parameters are calibrated in this section.

The calibration is done in two steps. First, the optical axis of the camera is aligned to a specific direction and second, several object points with known coordinates are shown to the camera to find out their image coordinates. With these data, Xc, Yc and F can be computed.

A wooden calibration frame was constructed and used to align the optical axis. This frame has three axes which are perpendicular to one another just like a Cartesian CS. Five tiny light sources were attached at different locations as shown in Fig. 4.2. To align the optical axis of the camera to the Z axis of the frame, the camera is placed in front of the calibration frame and all light sources are lit up. The location and orientation of the camera is then gradually adjusted and the pattern

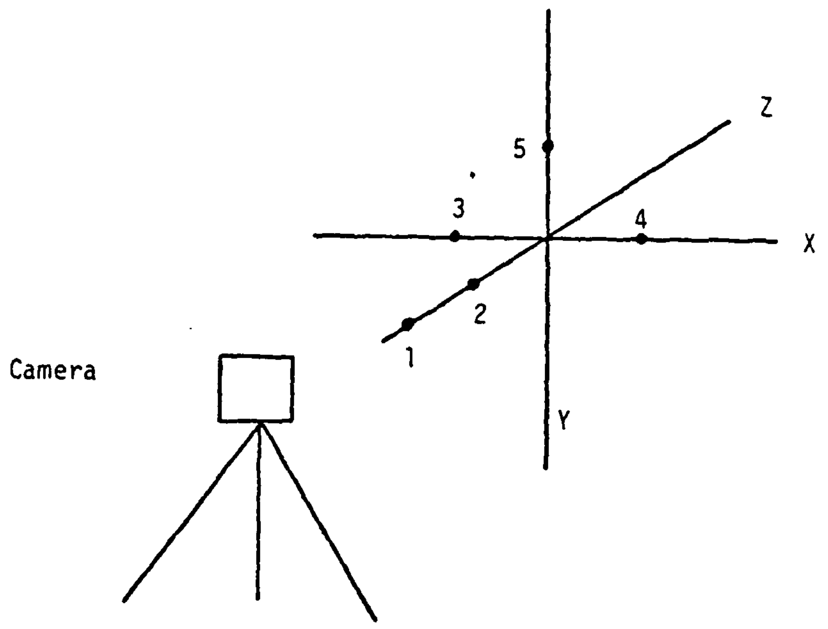


Figure 4.2 The Calibration Frame.

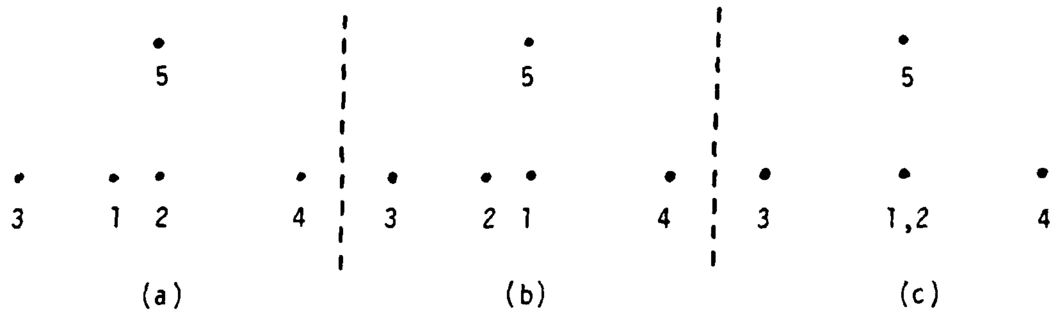


Figure 4.3 Image Patterns for Aligning the Optical Axis of a Camera.

(a) and (b) are Out of Alignment; (c) is In Alignment.

of the image is observed on a monitor. If the camera is out of alignment, five images will be seen on the screen as shown in Fig. 4.3.a and 4.3.b. If the camera is aligned, only four images are observed because the images of light sources 1 and 2 are overlapped, as shown in Fig. 4.3.c. A surface level was attached on top of each camera as another means to ensure the pitch angle is zero during calibration.

After alignment is achieved, the coordinates (Xi,Yi,Zi) of each light source are measured manually. Then the image coordinates (IXi,IYi) are obtained from the interface circuit. Thus, a 5-tuple (Xi,Yi,Zi,IXi,IYi) is generated for each light source. From IXi and IYi, the coordinates of the image point in camera CS can be found as:

$$Xpi = Dx \cdot (IXi-1/2) \tag{4.16}$$

$$Ypi = Dy \cdot (IYi-1/2) \tag{4.17}$$

$$Zpi = 0 \tag{4.18}$$

where Dx=46 μm and Dy=36 μm.

For each light source, two equations in the form of Eqs. (4.5) and (4.6) are obtained. They are

$$X_{pi} = \frac{(Zi/F) \cdot Xc+Xi}{1+(Zi/F)} \tag{4.19}$$

$$Y_{pi} = \frac{(Zi/F) \cdot Yc+Yi}{1+(Zi/F)} \tag{4.20}$$

Rearranging Eqs. (4.19) and (4.20) yields

$$Zi \cdot Xc + F \cdot (Xi - Xpi) = Zi \cdot Xpi \tag{4.21}$$

$$Zi \cdot Yc + F \cdot (Yi - Ypi) = Zi \cdot Ypi \tag{4.22}$$

It is clear that with the data of any two light sources (except 1 and 2 since they lie on the same line), four equations in the form of Eqs. (4.21) and (4.22) may be derived and the three unknowns Xc, Yc and F can be solved. A program, CENTER, has been written to take data from two light sources and return the values of Xc, Yc and F.

Each of the two cameras was calibrated separately, and they did exhibit different values of Xc and Yc. One important observation was that the image of the optical axis, a point where the optical axis intersects the image plane, does not coincide with the center of the image sensor. The image sensor has 244 rows and 248 columns hence the image coordinates of the center of the image sensor should be (124,122). However, it was found that the optical axis of camera 1 intersects the image plane at (119,118). For camera 2 , that location is at (111, 113).

4.4 Lens Calibration

The existence of image distortion can be verified in at least two ways. In one case, a picture containing the junction line of a wall and the floor was displayed on a monitor. Physically it is a straight line, but when observed on the screen, it became a curve, as shown in Fig. 4.4. In another test, eight light sources were placed in a horizontal line with the distance between any two neighbors being equal. Then the

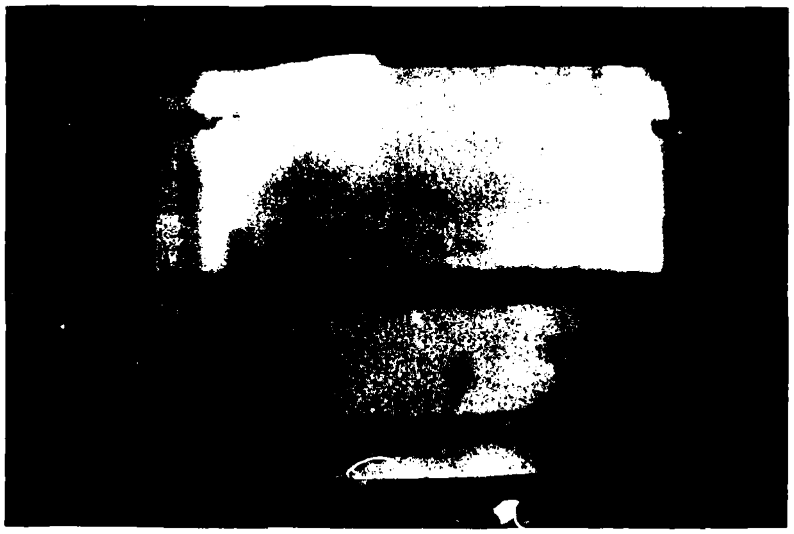


Figure 4.4 Image Distortion Caused by a Wide-Angle Lens.

image coordinates of each light were taken and compared. It was found that the difference in image coordinate of two neighbors was smaller at both ends.

To deal with the distortion caused by the wide-angle lenses, one has to know the type of distortion, and whether it can be expressed analytically and corrected.

Two types of distortion, barrel distortion and pincushion distortion [52], are the most common distortions that may appear in an optical system. Their patterns of distortion are shown in Fig. 4.5. Comparing Fig. 4.4 with Fig. 4.5, it is apparent that the distortion exists in cameras used in this research is not of the pincushion type but closer to the barrel distortion.

For barrel distortion, the amount of distortion is proportional to the cube of the radius of an image [52]. The radius of an image is defined as the distance between the image and the image of the optical axis. Although the observed distortion may not be a pure barrel distortion, it is of similar nature. Thus it was considered reasonable to express the distortion ΔR as a polynomial function of radius R, i.e.;

$$\Delta R = K_n R^n + K_{n-1} R^{n-1} + \dots + K_1 R + K_0 \qquad (4.23)$$

To collect distortion data, square-ruled papers were posted on a blackboard. A light source was manually moved from one location to another location. At each position, a 5-tuple (Xi,Yi,Zi,IXi,IYi) was obtained as mentioned earlier. Using Eqs. (4.19) and (4.20), one can find the distortionless image (Xpi,Ypi). The distance between this point and the image of the optical axis is the non-distorted radius Ri.

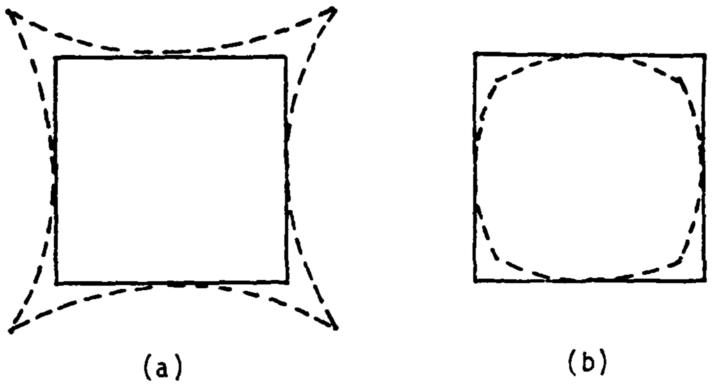


Figure 4.5 Common Distortion Patterns,
(a) Pin-Cushion Distortion;
(b) Barrel Distortion.

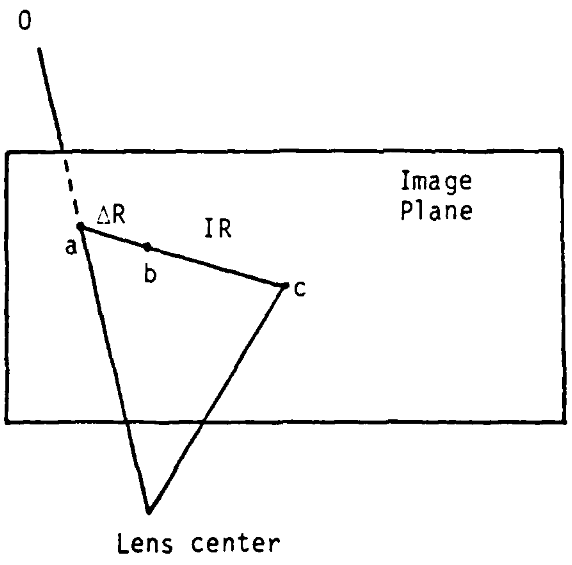


Figure 4.6 Relationships Between the Nondistorted Image a, the Distorted Image b, and the Image of the Optical Axis c.

The image detected by the interface circuit, (IXi,IYi), is a distorted image. Its radius is denoted as IRi. The amount of distortion ΔRi is the difference of Ri and IRi. This is as illustrated in Fig. 4.6.

$$\Delta Ri = Ri - IRi \tag{4.24}$$

Thus, a pair (ΔRi,Ri) is generated from each 5-tuple. Program CAC1 (listed in Appendix) reads a 5-tuple data from an external file, computes (ΔRi,Ri), and stores them into another data file.

The next step is to fit collected data with a polynomial in the form of Eq. (4.23). Two questions arise here with respect to the degree of the polynomial and the determination of the coefficients. The approach taken was to begin with the lowest degree possible (a first order polynomial), and increase the degree by one each time, meanwhile comparing the results of the curve fitting. The performance of an mth order polynomial is defined as

$$Sm = \sum_{i=1}^{n} \left| \Delta Ri - Pm(Ri) \right|^2 \tag{4.25}$$

where Pm is the mth order polynomial, n is the number of data points, and Sm is the performance index of Pm.

A program POLYFT has been created for the above purpose. Given the desired order m of the polynomial and the number of data points, it returns the value of Sm and all the coefficients. The coefficients are calculated by an algorithm based on a set of orthogonal polynomials [53], and the polynomial thus obtained is, among all mth degree polynomials, the one that best fits the data in the minimum-square-error sense. If Sm is significantly smaller than Sm-1, then polynomials of degree m+1 should be tried and so on. Otherwise, Pm-1 would be chosen a. the desired polynomial.

After going through the process just described, it was concluded that a 4th order polynomial should be used. The coefficients for camera 1 are

$$\begin{bmatrix} A4 \\ A3 \\ A2 \\ A1 \\ A0 \end{bmatrix} = \begin{bmatrix} -6.42E5 \\ 7.8779E3 \\ 1.6978E1 \\ 1.11458E-1 \\ 1.17956E-4 \end{bmatrix} \qquad (4.26)$$

Those for camera 2 are

$$\begin{bmatrix} B4 \\ B3 \\ B2 \\ B1 \\ B0 \end{bmatrix} = \begin{bmatrix} -6.432E5 \\ 8.9594E3 \\ 3.9178 \\ -6.7241E-2 \\ 8.38079E-5 \end{bmatrix} \qquad (4.27)$$

Given these coefficients, lens calibration can be stated as follows: given an distorted image(IX,IY), find the undistorted radius R which may in turn be used to locate the distortionless image. From (IX,IY), the quantity IR is computed. Combining Eq. (4.23) and (4.24) results in

$$\Delta R = R\text{-}IR = K_4R^4 + K_3\,R^3 + K_2R^2 + K_1R^2 + K_o \qquad (4.28)$$

Rearranging yields

$$K_4R^4 + K_3R^3 + K_2R^2 + (K_1-1)R + K_0 = IR \tag{4.29}$$

Therefore, by solving the real roots of a 4th order polynomial, the value of R is obtained. Since images are inwardly distorted, the undistorted radius R must be greater than the distorted radius IR. This condition can be used to choose the correct root in cases where there are more than one real, positive roots. A program ROOT4 (see Appendix) was written for this purpose using the Newton-Raphson method.

The lens calibration data are listed in Table 4.1 and Table 4.2. Here IXD and IYD are the distorted image coordinates furnished by the hardware circuit, IXUD and IYUD are the undistorted image coordinates computed using the camera model, and IXC and IYC are the compensated values obtained by using the lens calibration procedure described above. Table 4.1 is associated with Camera 1 and Table 4.2 is for Camera 2. Note that without correction, the distortion may be as large as 10 pixels, and after compensation, it is reduced to within 1 pixel.

In the above analysis, distortion ΔR is expressed as a polynomial function of the undistorted image radius R. It may also be possible to represent ΔR as a polynomial function of the distorted image radius IR. Then adding ΔR to IR would give the value of R. This approach is computationally more efficient because it is not necessary to solve for the real roots of a fourth order polynomial.

TABLE 4.1
Lens Calibration Data of Camera 1.

(IXD and IYD are distorted image coordinates; IXUD and IYUD are the undistorted coordinates; and IXC and IYC are the compensated values.)

	IXD	IYD	IXUD	IYUD	IXC	IYC
1	220	110	233	109	234	109
2	218	69	233	60	234	61
3	216	42	233	27	234	27
4	214	30	233	10	233	11
5	219	18	243	-6	243	-7
6	222	44	243	27	243	28
7	224	70	243	60	243	61
8	190	18	200	2	201	2
9	164	14	167	2	168	2
10	168	85	167	84	168	84
11	136	51	135	51	135	50
12	136	68	135	68	135	68
13	136	84	135	84	135	84
14	103	83	102	84	102	85
15	76	12	69	2	70	2
16	49	16	36	2	36	1
17	47	29	36	19	36	18
18	19	16	-6	-6	-7	-8
19	15	54	-6	43	-7	43
20	13	81	-6	76	-7	76

TABLE 4.2

Lens Calibration Data of Camera 2.

(IXD and IYD are distorted image coordinates; IXUD and IYUD are the undistorted coordinates; and IXC and IYC are the compensated values.)

	IXD	IYD	IXUD	IYUD	IXC	IYC
1	195	15	210	-3	210	-4
2	196	28	210	14	209	14
3	218	33	242	14	242	14
4	220	45	242	30	243	30
5	221	58	242	46	243	46
6	223	99	242	96	243	96
7	223	113	242	112	242	113
8	172	23	177	14	178	13
9	176	97	177	96	178	96
10	143	20	144	14	144	13
11	112	47	111	46	111	46
12	54	15	46	5	46	3
13	50	72	46	71	46	70
14	83	8	79	-3	79	-2
15	68	8	62	-3	62	-4

4.5 Triangulation

It is clear that every object point has a single, well defined image point. Thus, given an object point O, the corresponding image point V on the image plane is uniquely defined. The inverse mapping, however, is not one-to-one. For each image point there is a line in space, defined by the image point and the lens center, along which the corresponding object point must lie. Additional information is needed to determine the exact location of the object point. The standard method for this purpose is called stereoscopy [54], and is based upon the use of two pictures.

4.5.1 Mathematical Model Of Triangulation

For stereoscopic analysis, a binocular system can be represented as in Fig. 4.7. Here, O is the object point to be located, I1 and I2 are the image points on the two image planes, C1 and C2 are the lens centers and D is the displacement vector between C1 and C2. Knowing I1 and C1, a vector $\overline{R1}$ is determined. Similarly, another vector $\overline{R2}$ is defined by C2 and I2. The intersection of $\overline{R1}$ and $\overline{R2}$ gives the location of O. Mathematically, this can be written as

$$\overline{R1} = \overline{D} + \overline{R2} \tag{4.30}$$

Let $\overline{U1}$ and $\overline{U2}$ be the unit vectors along $\overline{R1}$ and $\overline{R2}$, Eq. (4.30) becomes

$$a\overline{U1} = \overline{D} + b\overline{U2} \tag{4.31}$$

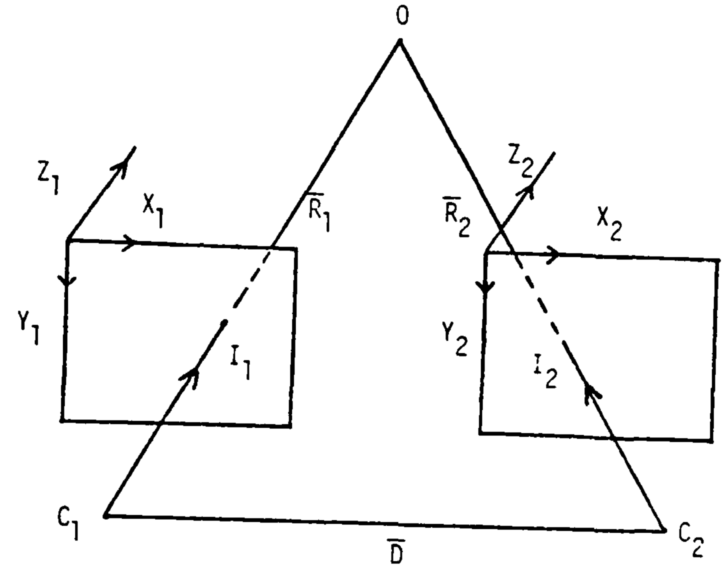


Figure 4.7 Triangulation using Two Cameras.

where a and b are constants.

Ideally, $\overline{R1}$ and $\overline{R2}$ should intersect at a unique point. However, due to various errors, the two projection rays will in general fail to intersect. A reasonable solution, proposed by Duda and Hart [54], is to place $\overline{O}$ midway between $\overline{R1}$ and $\overline{R2}$ where the distance between them is minimum, i.e.,

$$\overline{O} = [a_0\overline{U1} + (\overline{D}+b_0\overline{U2})]/2 \tag{4.32}$$

where a_0 and b_0 are the values of a and b that minimize

$$J(a,b) = \left\| a\overline{U1} - (\overline{D}+b\overline{U2}) \right\| \tag{4.33}$$

To solve for a_0 and b_0, rewrite J(a,b) as

$$J(a,b) = [a\overline{U1}-(\overline{D}+b\overline{U2})]\cdot[a\overline{U1}-(\overline{D}+b\overline{U2})] \tag{4.34}$$

Expanding Eq. (4.34) yields

$$J(a,b)=a^2 - 2a\overline{U_1}\cdot\overline{D} - 2ab\overline{U_1}\cdot\overline{U_2} + 2b\overline{U_2}\cdot\overline{D} + \overline{D}^2 + b^2 \tag{4.35}$$

The conditions for minimizing J(a,b) are

$$dJ/da=0$$
$$dJ/db=0 \tag{4.36}$$

which lead to

$$2a - 2\overline{U_1} \cdot (\overline{D} + b\overline{U_2}) = 0 \tag{4.37}$$

$$2b - 2\overline{U_2} \cdot (\overline{D} - a\overline{U_1}) = 0 \tag{4.38}$$

Substituting Eq. (4.37) into Eq. (4.38) and eliminating the variable a gives

$$b + \overline{U_2} \cdot \overline{D} - (\overline{U_1}\cdot\overline{U_2})(\overline{U_1}\cdot\overline{D} + b\overline{U_1}\cdot\overline{U_2}) = 0 \tag{4.39}$$

Solving for b results in

$$b_0 = \frac{(\overline{U_1}\cdot\overline{U_2})(\overline{U_1}\ \overline{D}) - (\overline{U_2}\cdot\overline{D})}{1 - (\overline{U_1}\cdot\overline{U_2})^2} \tag{4.40}$$

Similarly,

$$a_0 = \frac{(\overline{U_1}\cdot\overline{D}) - (\overline{U_1}\cdot\overline{U_2})(\overline{U_2}\cdot\overline{D})}{1 - (\overline{U_1} \cdot \overline{U_2})^2} \tag{4.41}$$

Finally, the position of O is

$$\overline{O} = a_0\cdot\overline{U_1} \tag{4.42}$$

or

$$\overline{O} = \overline{D} + b_0\cdot\overline{U_2} \tag{4.43}$$

4.5.2 Physical Considerations

In the above analysis, all positions and vectors are represented in coordinate-free notations. For the actual arrangement of a binocular system, two CS's are involved, one for each camera. Information such as image coordinates (I1 and I2) and locations of lens centers (C1 and C2) are expressed in two different CS's. Before applying the results from 4.5.1, it is necessary to define a common CS and refer everything to it. Here, camera 1 CS is chosen as the reference CS (camera 2 CS can be used as well). Furthermore, the geometric relation between the two cameras must be known so that entities in the camera 2 CS can be transformed to the camera 1 CS. This relation may be represented as a HTM (Homogeneous Transformation Matrix) [55]. A HTM which describes the spatial relation of two arbitrary CS's A and B is a 4 by 4 matrix denoted as ${}^{b}T_{a}$, and the elements of this matrix are defined as follows:

$$ {}^{b}T_{a} = \begin{bmatrix} t11 & t12 & t13 & t14 \\ t21 & t22 & t23 & t24 \\ t31 & t32 & t33 & t34 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.44) $$

where (t14,t24,t34) is the representation of the origin of the A CS in the B CS. The column (t11,t21,t31) is the representation of the unit X vector of the A CS in the B CS. Similarly, (t12,t22,t32) and (t13,t23,t33) are the representations of the unit Y vector and the unit Z vector of the A CS in the B CS. The 9 elements (t11,t12,...,t33) specify the "orientation" of the A CS with respect to the B CS, while the 3 elements (t14,t24,t34) contain the "displacement" information. For example, ${}^{b}T_{a}$ for CS's A and B in Fig. 4.8 is

$$ {}^{b}T_{a} = \begin{bmatrix} 0 & 0 & -1 & 3 \\ 1 & 0 & 0 & 4 \\ 0 & -1 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.45) $$

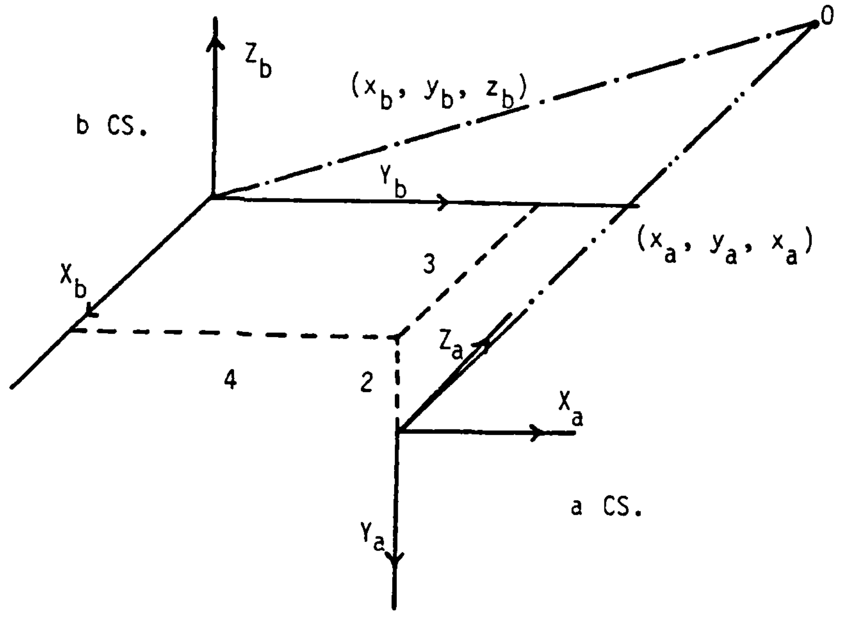


Figure 4.8 Relationship Between Coordinate Systems a and b.

By means of this HTM, the coordinates of any point (Xa,Ya,Za) in the a CS can be transformed into the representation in the b CS (Xb,Yb,Zb) as

$$\begin{bmatrix} Xb \\ Yb \\ Zb \\ 1 \end{bmatrix} = {}^{b}T_{a} \cdot \begin{bmatrix} Xa \\ Ya \\ Za \\ 1 \end{bmatrix} \qquad (4.46)$$

An important characteristic of a HMT ${}^{b}T_{a}$ is that

$${}^{a}T_{b} = ({}^{b}T_{a})^{-1} \qquad (4.47)$$

Suppose that the image coordinates of I1 are (IX1, IY1). Then its 3-D coordinates as expressed in camera 1 CS are (X1,Y1,Z1) where

$$X1=Dx\cdot(IX1-1/2), \qquad (4.48)$$

$$Y1=Dy\cdot(IY1-1/2), \qquad (4.49)$$

$$Z1=0 \qquad (4.50)$$

The coordinates of the lens center C1 are (XC1,YC1,ZC1). The components of vector R1, (RX1,RY1,RZ1), are simply

$$RX1=X1-XC1, \qquad (4.51)$$

$$RY1=Y1-YC1, \qquad (4.52)$$

$$RZ1=Z1-ZC1. \qquad (4.53)$$

Similarly, the coordinates of I2 and C2 are (X2,Y2,Z2) and (XC2,YC2, ZC2), respectively, as expressed in the camera 2 CS. These entities are transformed to camera 1 CS by using HTM ${}^{1}T_{2}$, i.e.;

$$I2' = {}^{1}T_{2} \cdot I2 \quad , \tag{4.54}$$

$$C2' = {}^{1}T_{2} \cdot C2 \tag{4.55}$$

$$R2 = I2' - C2'. \tag{4.56}$$

The unit vectors $\overline{U1}$ and $\overline{U2}$ are then computed as follows

$$\overline{U1} = \overline{R1}/\left|\overline{R1}\right|,$$
$$\overline{U2} = \overline{R2}/\left|\overline{R2}\right|. \tag{4.57}$$

The displacement vector D is the same as the 4th column vector of ${}^{1}T_{2}$ but without the last element 1. That is,

$$DX = {}^{1}T_{2}(4,1),$$

$$Dy = {}^{1}T_{2}(4,2)$$

$$DZ = {}^{1}T_{2}(4,3). \tag{4.58}$$

Now with all vectors computed and expressed in the camera 1 CS, one can substitute them into Eqs. (4.40) and (4.41) and find the location of O. The described triangulation is done by the program TRIANG which takes ${}^{1}T_{2}$, IX1,IY1,IX2 and IY2 as input data and returns the coordinates of O in camera 1 CS.

4.6 Camera Installation Arrangement

The cameras are installed onto the Hexapod by using a π -shaped frame, designed and built by the Mechnical Engineering Department, and two KALT #5020 ball-socket heads (Fig. 4.9). A drawing showing the

Figure 4.9 Picture of the Ball-Socket Heads.

frame in relation to the Hexapod is given in Fig. 4.10. A picture showing the physical mounting is given in Fig. 4.11. The cameras are mounted directly to the ball-socket heads, which are then attached to the frame through two mechanical adaptors (G and H). The frame is installed onto the Hexapod by screwing the bases BJ and DK tightly to the existing structure so that it is rigid and stable. Adaptors G and H can move along segment EF, thus the separation between the cameras is adjustable. Segment EF can slide up and down along segments AB and CD, hence the altitude of both cameras is subject to change. These provide sufficient flexbility in choosing suitable locations for the cameras. Moreover, the ball-socket head allows a camera to be tilted and rolled, and therefore adds two more degrees of freedom.

4.6.1 Altitude Of Cameras

With a 6.5 mm wide-angle lens, a camera can posess a 2 m wide field of view at an altitude of 1.2 m above ground. Although the structure allows a maximum height of 1.5 m and a larger working distance would provide wider field of view, the received intensity of an image becomes weaker as a camera is installed higher thus the image is more likely to be skipped by the image decoder. Also, lower camera mounting would give better mechanical stability. Hence, it is advantageous to keep the altitude of cameras lower so long as the field of view is large enough to cover the area of interest.

4.6.2 Separation Between Cameras

The distance between the cameras will affect both the CFOV (Common Field Of View) and the accuracy of triangulation. A large separation is

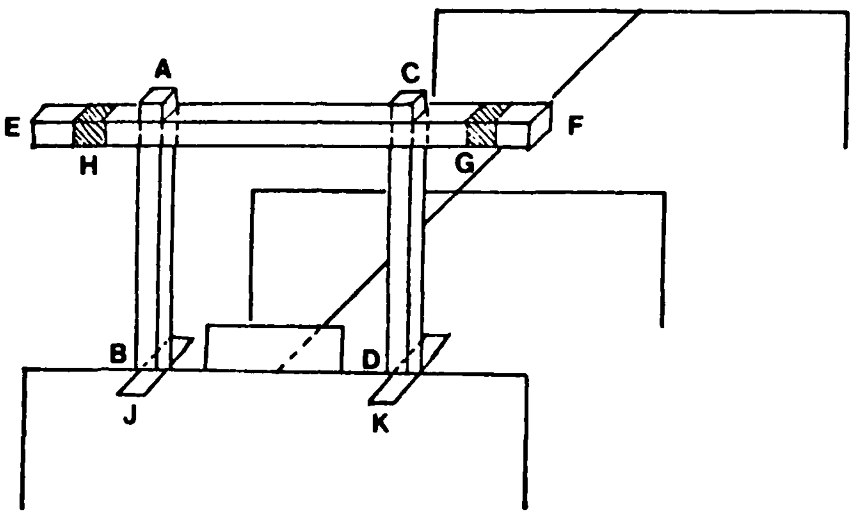


Figure 4.10 Camera Mounting Frame and the Body of the Hexapod.

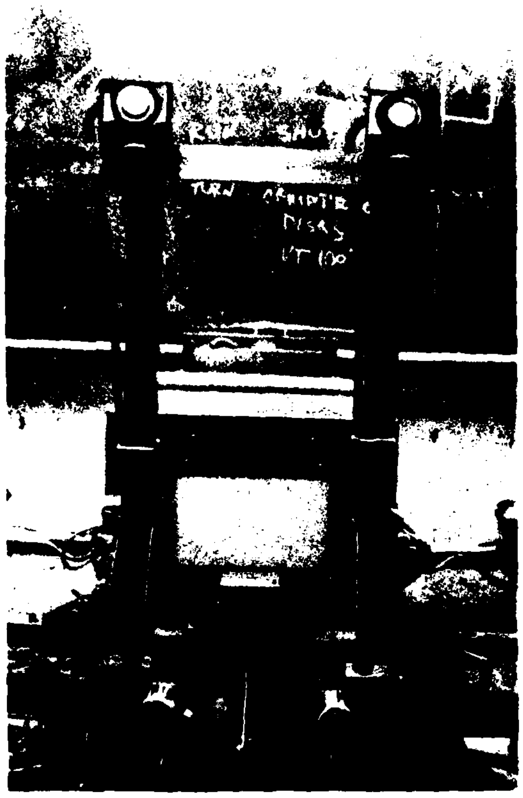


Figure 4.11 Picture Showing the Physical Camera Mounting.

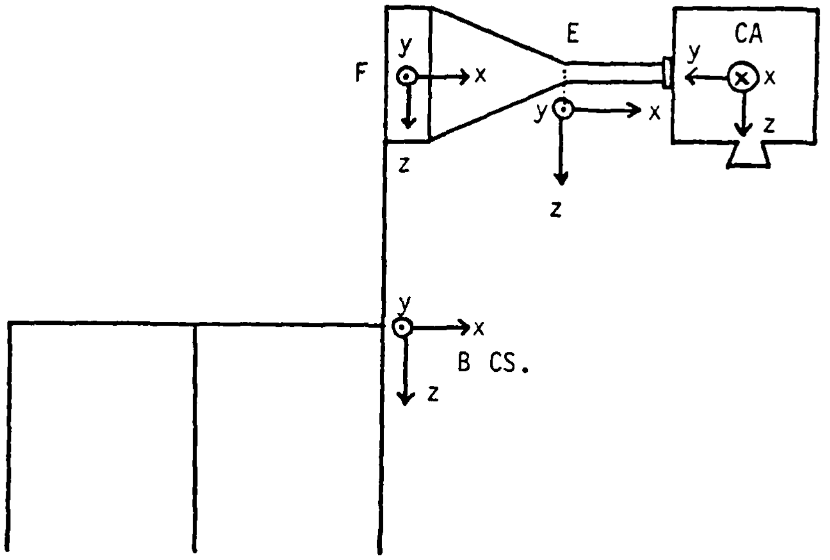
desirable so that triangulation gives accurate result [37]. A smaller separation, however, permits better CFOV. A compromise value, 40 cm, was picked after a number of experiments.

4.6.3 Tilt And Roll Angles

A judicious selection of a tilt angle and a roll angle will produce a better CFOV. The following analysis will verify this point. It involves a series of coordinate transformation from the camera CS to the body CS. Two intermediate CS's E and F are introduced to facilitate the explanation. Fig. 4.12 shows a case in which both tilt and roll angles are zero. The sizes of the camera and the ball-socket are exaggerated to make the picture clear. The origin of the E CS corresponds to the junction point of the moveable segment and the cone-shaped base. The X axis is assigned to be along the direction of the segment, the Z axis is perpendicular to the ground, and the Y axis is determined by the right-hand rule. The origin of the F CS is at the center of the base of the ball-socket head. The X axis is perpendicular to the base, the Z axis is perpendicular to the ground, and again the Y axis follows the right-hand rule. The camera CS, denoted as CA, is as described in Section 4.2.

Regardless of the tilt and roll angles, the CA and E CS's are related by a HTM

$$ {}^{E}T_{ca} = \begin{bmatrix} 0 & -1 & 0 & 1x \\ 1 & 0 & 0 & 1y \\ 0 & 0 & 1 & 1z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.59) $$



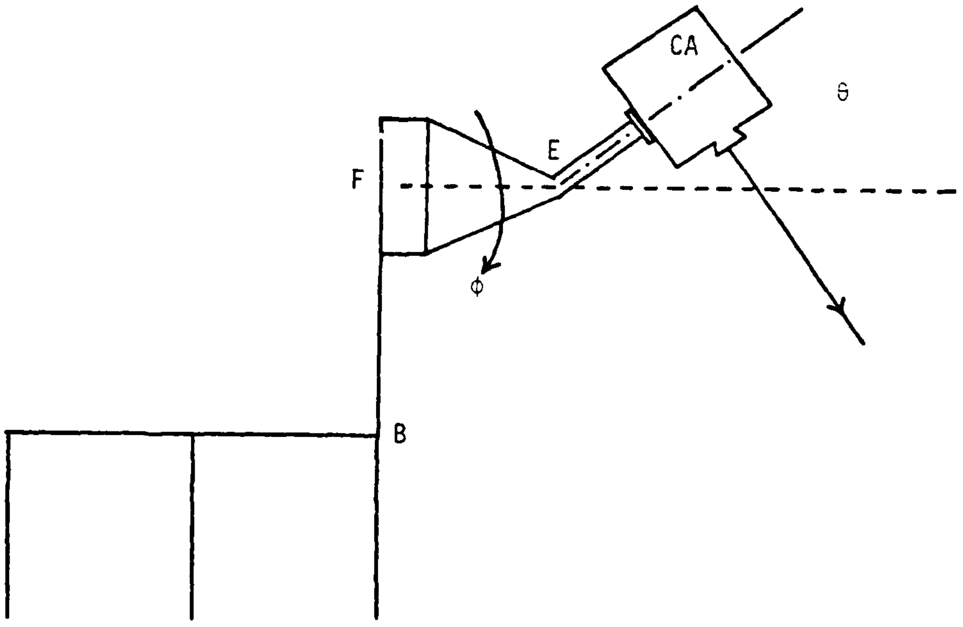


Figure 4.12 Relationships Between Cameras and Hexapod Body in Terms of Pitch and Roll Angles.

To tilt the camera for an angle of θ, the segment is rotated around the Y axis for an angle of θ. At this position, the E and F CS's are related by a second HTM

$$ {}^{B}T_{F} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & px \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.60} $$

Then, if the camera is rolled by an angle of ϕ by rotating the base about its X axis, the F and body CS's can be related by a third HTM

$$ {}^{B}T_{F} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & \cos\phi & -\sin\phi & dy \\ 0 & \sin\phi & \cos\phi & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.61} $$

Multiplying these matrices, the relation between the CA and body CS's is given by

$$ {}^{B}T_{CA} = {}^{B}T_{F} \cdot {}^{F}T_{E} \cdot {}^{E}T_{CA} $$

$$ {}^{B}T_{CA} = \begin{bmatrix} 0 & -\cos\theta & \sin\theta & lx\cos\theta + lz\sin\theta + dx + px \\ \cos\phi & -\sin\theta\sin\phi & -\sin\phi\cos\theta & lx\sin\phi\sin\theta + ly\cos\phi - lz\sin\phi\cos\theta + dy \\ \sin\phi & \sin\theta\cos\phi & \cos\phi\cos\theta & -lx\sin\theta\cos\phi + ly\sin\phi + lz\cos\theta\cos\phi + lz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.62} $$

Since the goal is to find the field of view as seen from the body CS, it is appropriate to explain here how a field of view is determined. A field of view is formed by the four intersection points of four vectors with ground. These four vectors all originate from the lens center but pass through a different corner of the image sensor, as shown in Fig. 4.13. Knowing the 3-D coordinates of the lens center and the four corners in CA CS, these vectors can be computed. By using HTM T, these vectors can be expressed in the body CS. The ground, when represented in the body CS, is a plane characterized by an equation Z=k, with k depending on the altitude of the body. By solving for the intersections of four lines and one plane, four points are obtained, and the trapezoid defined by these points is the field of view. The following example illustrates the concept outlined above by finding the point of intersection of the vector C4 and the ground. The other three vectors can be treated in exactly the same way.

The coordinates of the lens center are (Xc,Yc,Zc), and those of corner 4 are (X4,Y4,Z4) with

$$X4 = 248 \cdot Dx,$$
$$Y4 = 244 \cdot Dy,$$
$$Z4 = 0 \qquad (4.63)$$

These coordinates are transformed into the body CS by

$$\begin{bmatrix} X_4' \\ Y_4' \\ Z_4' \\ 1 \end{bmatrix} = {}^{B}T_{CA} \cdot \begin{bmatrix} X_4 \\ Y_4 \\ Z_4 \\ 1 \end{bmatrix}, \qquad \begin{bmatrix} X_c' \\ Y_c' \\ Z_c' \\ 1 \end{bmatrix} = {}^{B}T_{CA} \cdot \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \qquad (4.64)$$

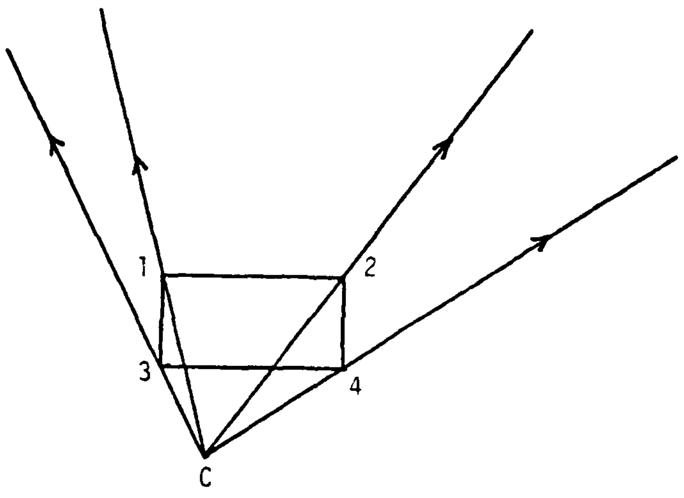


Figure 4.13 The Four Vectors Determining the Field of View.

The line that passes through C and corner 4 can be described by the following parametric equations:

$$X=X4'+m(X4'-Xc')$$
$$Y=Y4'+m(Y4'-Yc')$$
$$Z=Z4'+m(Z4'-Zc') \quad (4.65)$$

Recall that the ground is a plane with Z=k, which means that for the intersection point

$$k=Z4'+m(Z4'-Zc') \quad (4.66)$$

thus the unknown $m=(k-Z4')/(Z4'-Zc')$ and the value of X,Y and Z are now computed.

A program, FOVF, is available which generates the coordinates of the four vertices of the field of view for given tilt and roll angles. The resultant field of view is displayed on the HP graphic terminal so that one can easily see if it covers the area of interest. Fig. 4.14 through Fig. 4.17 are typical results generated by FOVF. Both tilt and roll angles are zero for the case of Fig. 4.14, which means that both cameras look straight down to the ground. Note that leg 1 is barely covered by the camera on the right-hand side, thus it may miss a foothold that is to the left of leg 1. Also note that the front (and rear) edges of the two field of views fail to overlap each other. This is due to the non-ideal location of the lens center with respect to the focal plane, as discussed in Section 4.3. Fig. 4.15 is the result produced by a tilt angle of 20 degrees and a roll angle of 0 degree.

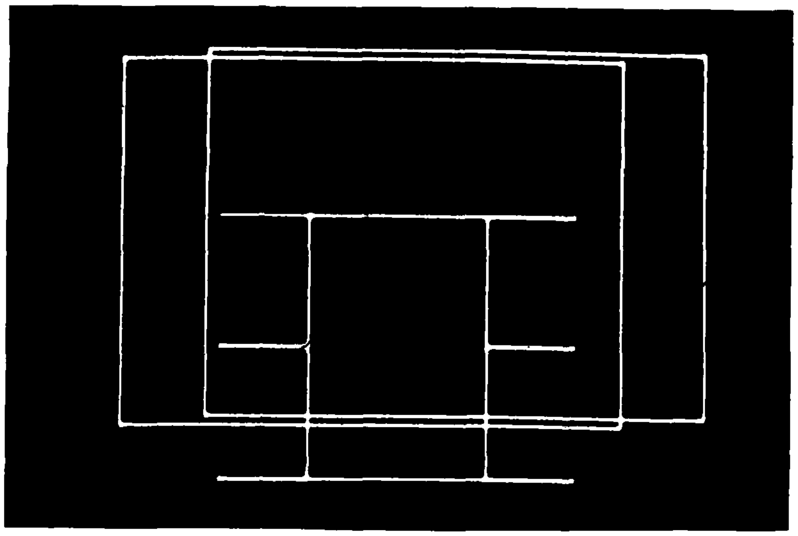


Figure 4.14 CFOV with a Tilt Angle of 0 degree and a Roll Angle of 0 degree.

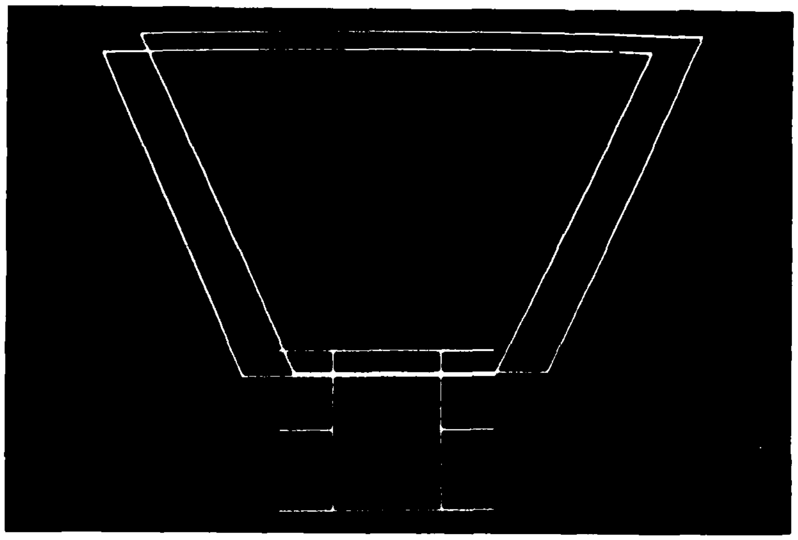


Figure 4.15 CFOV with a Tilt Angle of 20 degrees and a Roll Angle of 0 degree.

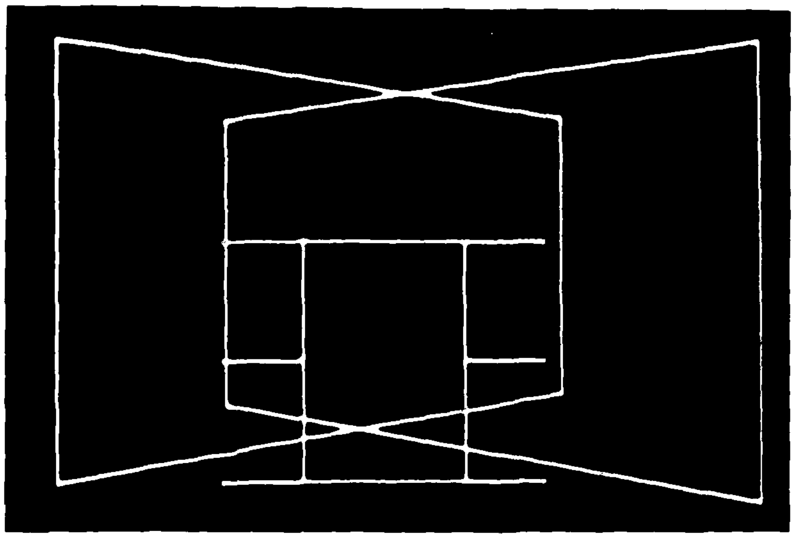


Figure 4.16 CFOV with a Tilt Angle of 0 degree and a Roll Angle of 20 degrees.

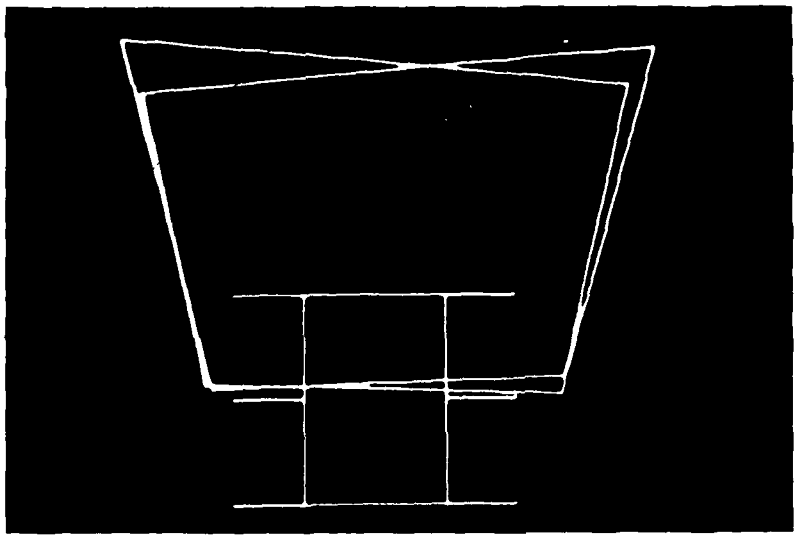


Figure 4.17 CFOV with a Tilt Angle of 20 degrees and a Roll Angle of 5 degrees.

The field of view of both cameras are tilted forward, and the coverage of the area of interest does not improve. For the case of a pure roll angle, as shown in Fig. 4.16 where the tilt angle is zero and the roll angle is 20 degrees, the result is not satisfactory either in terms of the coverage of the area of interest. With a judicious choice of tilt and roll angles, however, the CFOV can be greatly improved as seen in Fig. 4.17, where the tilt angle is 20 degrees and the roll angle is 5 degrees.

4.7 Geometrical Relationships Between Cameras And The Hexapod Body

After some experiments, it was observed that the ball-socket head was not rigid enough to lock itself to the selected pitch and roll angles. Thus the geometrical relationship between the cameras and the body could drift away from what it should be, and an error is introduced as the coordinates of a foothold are transformed from the camera CS to the body CS. To eliminate this problem, the ball-socket head was replaced by fixtures which allow the cameras to be more tightly attached to the mounting frame. Fig. 4.18 shows the top view of this arrangement. Fig. 4.19.a shows that Camera 1 being attached to the right fixture, and Fig. 4.19.b shows the physical connection between Camera 2 and the left fixture. With the new fixtures, the relationships between cameras and body are different from those when ball-socket heads were used and must be reidentified.

The geometrical relation between two CS' s can be decomposed into two parts: position and orientation. In an HTM, orientation is represented by the upper-left 3 x 3 matrix and position is treated as a vector in the rightmost column. Relating to the physical setup,

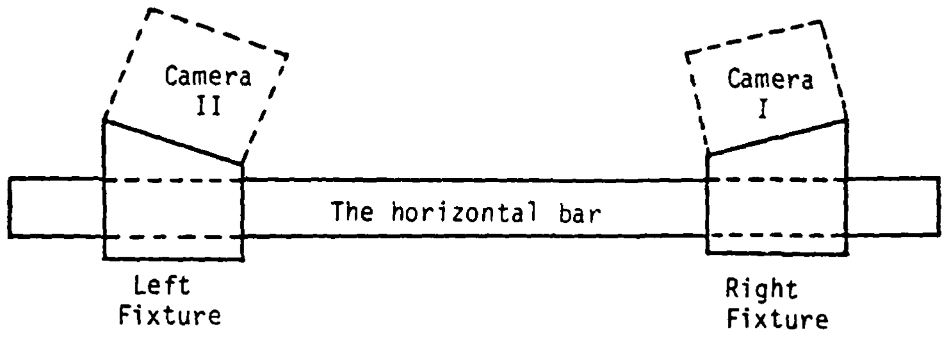
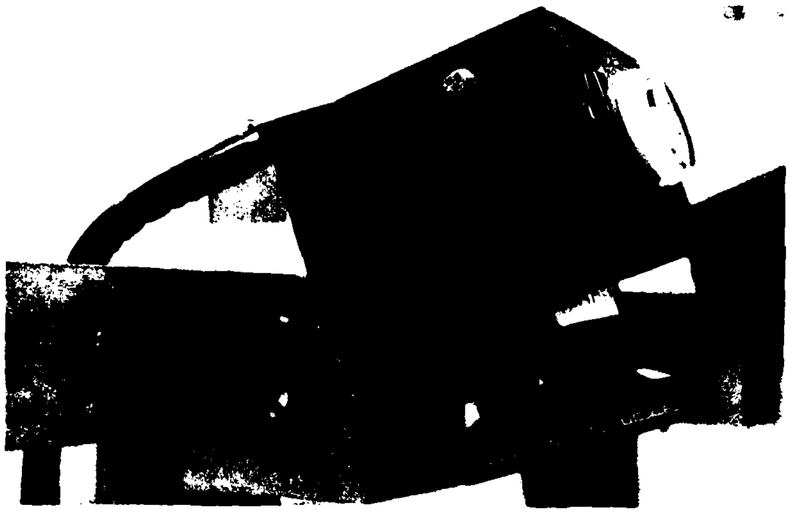
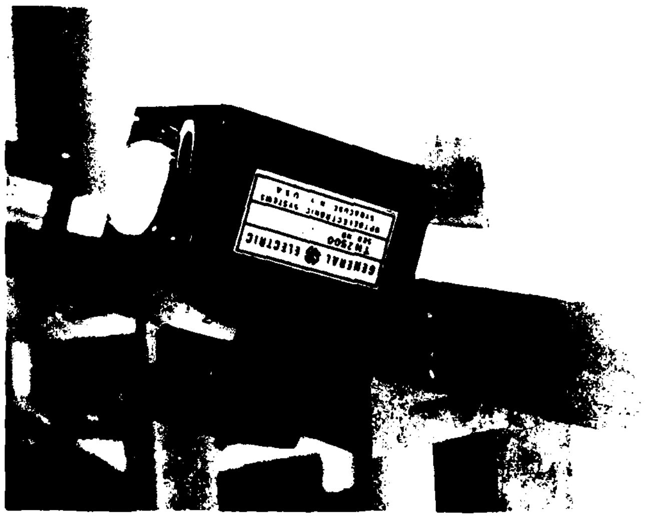


Figure 4.18 Top View of the Cameras, Fixtures, and the Horizontal Bar.



(a)



(b)

Figure 4.19 Pictures Showing Cameras Being Attached to Fixtures. (a) Camera 1 and the Right Fixture; (b) Camera 2 and the Left Fixture.

the position component relates to where a fixture is placed on the frame. Changes in position can be made by sliding a fixture up and down, in and out. Since the fixtures were cut in a special way, there is only one possible orientation once a camera is fastened onto it. This unique orientation may be identified by examining the structure of the fixture. Fig. 4.20 shows the top and two side views of the right fixture. The left fixture is a mirror reflection of the right one.

To help the explanation, another CS, called the F (Fixture) CS, is introduced and has its origin at point 1 as shown in Fig. 4.20. Several observations which lead to the determination of the orientation of a camera with respect to the right fixture are as follows:

(1) the X axis of camera CS coincides with vector 42,

(2) the XZ plane of camera CS lies on the plane passing through points 1,2 and 3, (thus the negative Y axis of camera CS has the same direction as the normal vector of plane 123), and

(3) the Z axis is completely specified by the right hand rule once the X and Y axes are located.

The actual computation is as follows. First, the coordinates of points 1,2,3 and 4 are found to be (0,0,0), (1.1, 0, 3), (-0.45, -5, 0), and (0.6, -5, 3) respectively as expressed in the F CS. Vectors formed by these points are computed as:

$$\begin{aligned} \overline{V}_{12} &= (1.1,\ 0,\ 3) \\ \overline{V}_{23} &= (-0.45,\ -5,\ 0) \\ \overline{V}_{42} &= (0.5,\ 5,\ 0) \qquad (4.67) \end{aligned}$$

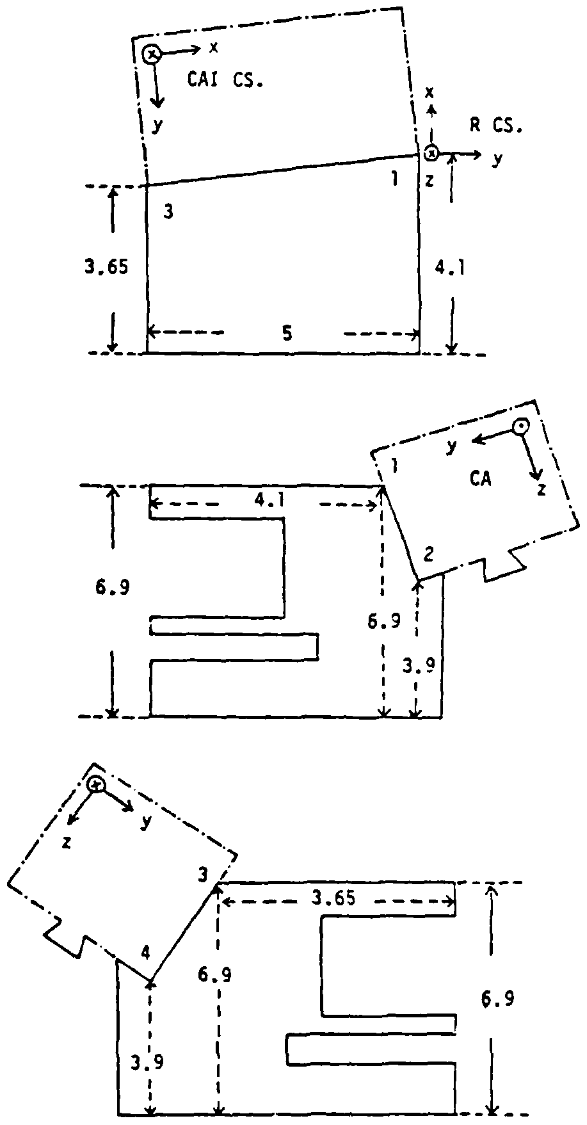


Figure 4.20 Geometrical Structure of the Right Fixture.

By observation (1), the representation of the X axis of camera CS in the F CS is the same as $\bar{V}_{42}$. Thus

$$\bar{X} = (0.5,\ 5,\ 0) \tag{4.68}$$

The cross product of $\bar{V}_{12}$ and $\bar{V}_{23}$ yields the normal vector of the plane 123, which may be computed by evaluting the determinant of the following 3x3 matrix:

$$\bar{V}_{12} \times \bar{V}_{23} = \begin{bmatrix} x & y & z \\ 1.1 & 0 & 3 \\ -0.45 & -5 & 0 \end{bmatrix} \tag{4.69}$$

The result is $\bar{N} = (15, -1.35, -5.5)$. After normalization, the normal vector $\bar{N}$ becomes (0.9355, -0.0842, -0.343). By observation (2), The Y axis of camera CS is represented as (-0.9355, 0.0842, 0.343). The Z axis is then computed as the cross product of $\bar{X}$ and $\bar{Y}$ as follows:

$$\bar{X} \times \bar{Y} = \begin{bmatrix} x & y & z \\ 0.0995 & 0.995 & 0 \\ -0.9355 & 0.0842 & 0.343 \end{bmatrix} \tag{4.70}$$

which gives $\bar{Z} = (0.341, -0.0341, 0.9392)$.

With $\bar{X}$, $\bar{Y}$ and $\bar{Z}$ computed as above, the orientation portion of ${}^{B}T_{CA1}$ is completely specified as shown below:

$${}^{B}T_{CA1} = \begin{bmatrix} 0.0995 & -0.9355 & 0.341 & px \\ 0.995 & 0.0842 & -0.0341 & py \\ 0 & 0.343 & 0.9392 & pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.71}$$

The position component (px,py,pz), which represents the coordinates of the origin of camera CS as seen from body CS, can usually be obtained by manual measurements. For instance, the position vector of camera 1 for the present installation is (-0.011, 0.14, -0.48) and that of camera 2 is (-0.011, -0.163, -0.48).

Therefore, ${}^{B}T_{CA1}$ is

$$ {}^{B}T_{CA1} = \begin{bmatrix} 0.0995 & -0.9355 & 0.341 & -0.011 \\ 0.995 & 0.0842 & -0.0341 & 0.14 \\ 0 & 0.343 & 0.9392 & -0.48 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.72) $$

The unit of all the numbers in ${}^{B}T_{CA1}$ is the meter.

Camera 2 was mounted to the frame by using the left fixture, whose structure is shown in Fig. 4.21. By going through the same derivation, ${}^{B}T_{CA2}$ can be identified as:

$$ {}^{B}T_{CA2} = \begin{bmatrix} 0.0995 & -0.9355 & 0.341 & -0.011 \\ 0.995 & -0.0842 & 0.0341 & -0.163 \\ 0 & 0.343 & 0.9392 & -0.48 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.73) $$

It should be emphasized again that, for both ${}^{B}T_{CA1}$ and ${}^{B}T_{CA2}$, the 3 x 3 orientation matrix is permanently fixed and will never change

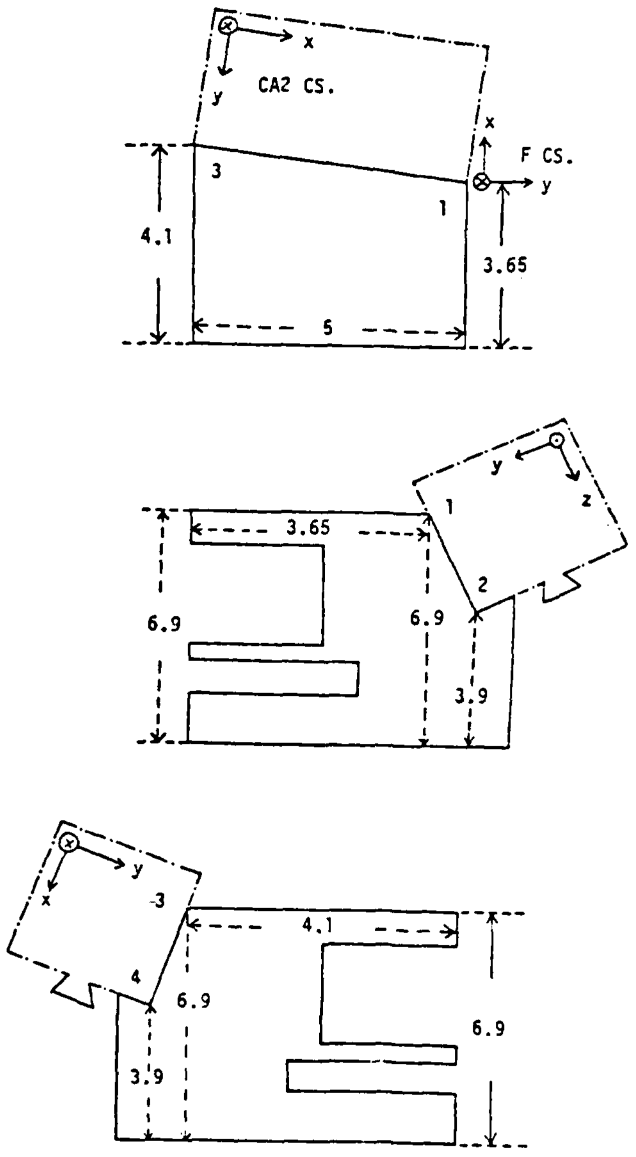


Figure 4.21 Geometrical Structure of the Left Fixture.

unless new fixtures with different structures are used. However, the position vector may vary as fixtures are moved around. Hence, it is necessary to measure px, py and pz whenever fixtures are relocated to new locations.

Fig. 4.22 shows a picture of the OSU Hexapod with the vision system installed on it.

4.8 Software Organization Of Triangulation

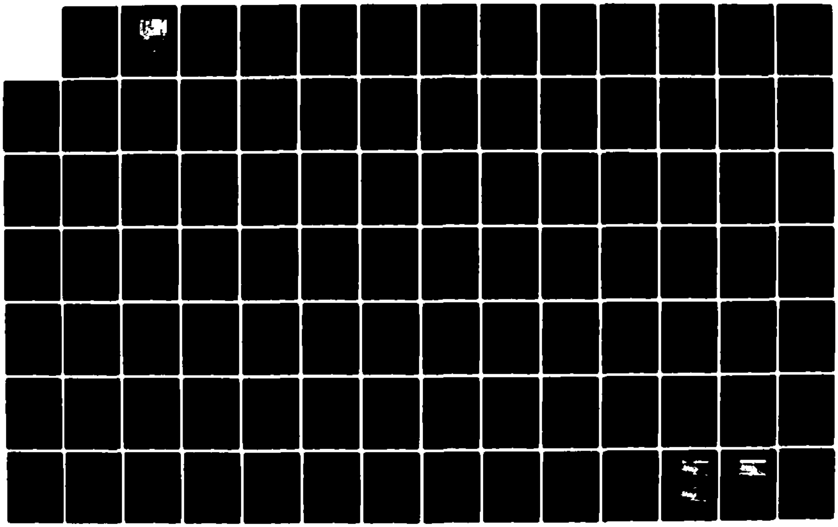
The conversion from image coordinates to three-dimensional spatial coordinates is summarized in a flowchart shown in Fig. 4.23. Image coordinates (IX1,IY1,IX2,IY2) are provided by the interface circuit and are brought to the computer by subroutine VID as described in Section 3.6. The distorted image radii IR1 and IR2 are computed using these data. Distortion compensation is then performed to find the non-distorted radius R1 and R2 by calling subroutine ROOT4. After the un-distorted images are located on both frames, projection rays $\overline{V1}$ and $\overline{V2}$ are constructed by using camera models. All quantities represented in the Camera 2 CS are transformed to the Camera 1 CS using HTM ${}^{1}T_{2}$. Specifically, $\overline{V2}$ is represented as $\overline{V2'}$ in Camera 1 CS. The intersection of $\overline{V1}$ and $\overline{V2'}$ gives the coordinates of the object point in Camera 1 CS. They are then transformed to the body CS through ${}^{B}T_{CA1}$.

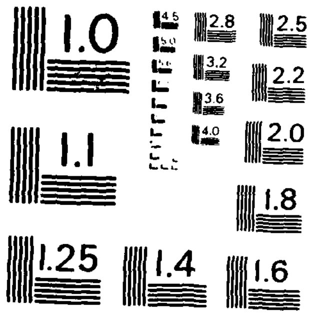
4.9 Summary

Four subjects were discussed in this chapter, including camera calibration, lens compensation, triangulation, and identification of the geometrical relationship between cameras and the Hexapod body.

AD-A131 449 AN EXPERIMENTAL STUDY OF AN ULTRA-MOBILE VEHICLE FOR OFF-ROAD TRANSPORTATION(U) OHIO STATE UNIV RESEARCH FOUNDATION COLUMBUS R B MCGHEE ET AL. JUL 83 3/4

UNCLASSIFIED MDA903-82-K-0058 F/G 6/4 NL




1.0
4.5
5.0
5.6
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

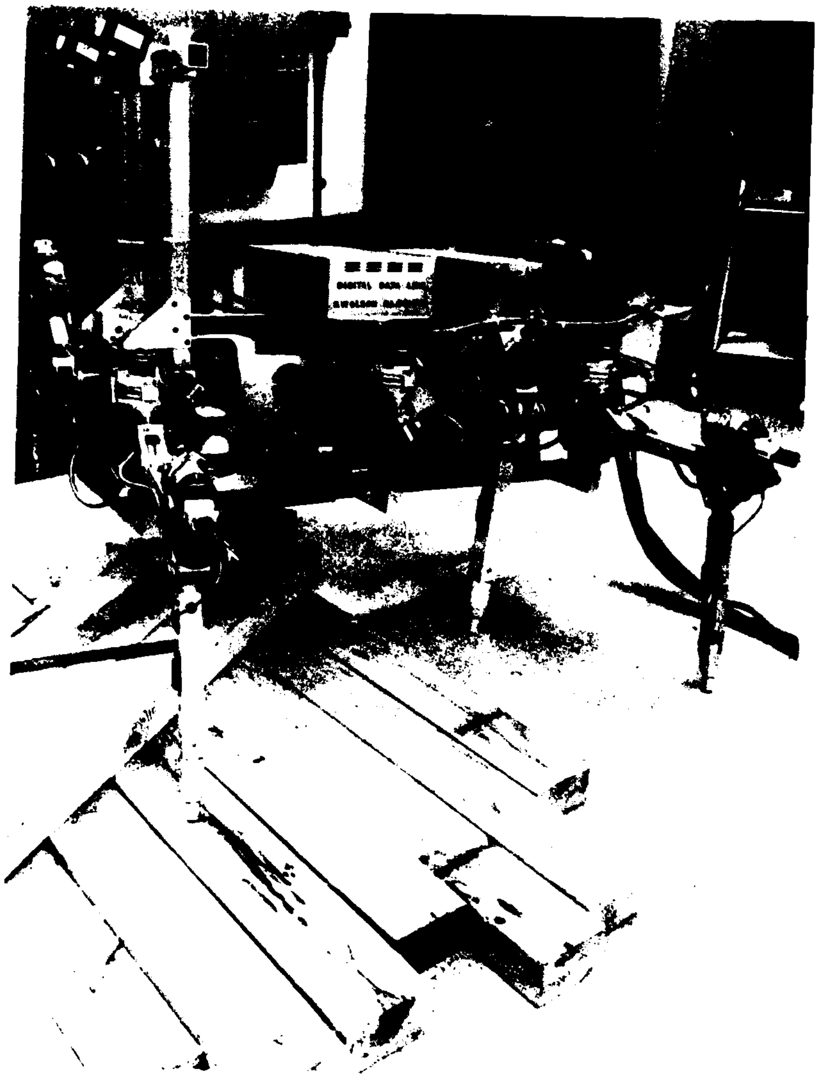


Figure 4.22 Picture of the OSU Hexapod After Installation of the Vision System.

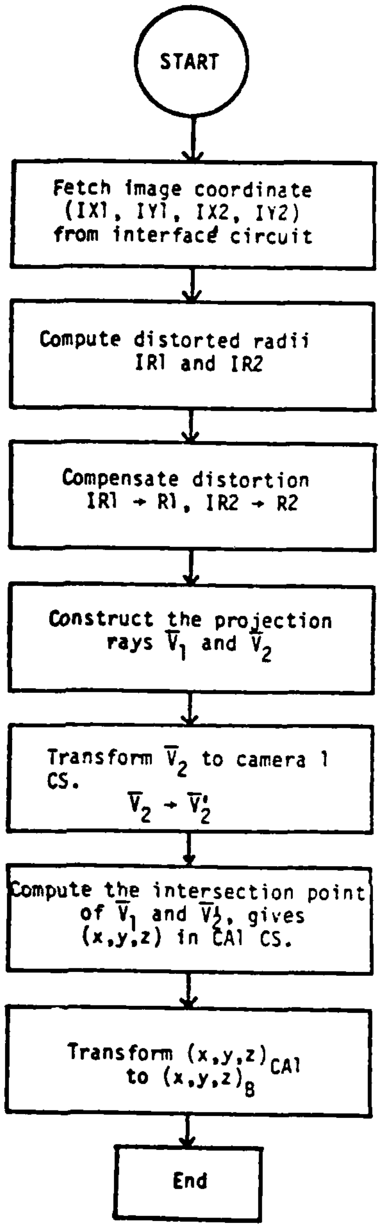

START
Fetch image coordinate (IX1, IY1, IX2, IY2) from interface circuit
Compute distorted radii IR1 and IR2
Compensate distortion IR1 → R1, IR2 → R2
Construct the projection rays $\overline{V}_1$ and $\overline{V}_2$
Transform $\overline{V}_2$ to camera 1 CS. $\overline{V}_2 \rightarrow \overline{V}_2'$
Compute the intersection point of $\overline{V}_1$ and $\overline{V}_2'$, gives (x,y,z) in CA1 CS.
Transform $(x,y,z)_{CA1}$ to $(x,y,z)_B$
End

Figure 4.23 Flowchart of Program TRIANG.

The camera model used in this work is a modified pin-hole model in which the focal plane is placed before the lens center. Camera calibration was performed to find the location of the lens center with respect to the focal plane. Ideally, the image coordinates of a lens center should be (124,122). However, it was found that the actually values are (119,118) and (111,113) for Camera 1 and Camera 2, respectively.

The wide-angle lenses introduced considerable image distortion, which was represented as a fourth order polynomial function of the undistorted image radius R. Lens compensation was performed on the distorted image coordinates so that the corrected image coordinates are approximately 1 pixel away from the undistorted values.

The three-dimensional coordinates of an object point were reconstructed from the compensated image coordinates of both cameras using triangulation method. Coordinate transformations were performed to bring all mathematical entities from Camera 2 CS to Camera 1 CS and from Camera 1 CS to body CS. The effect of tilt and roll angles on the common field of view was also presented.

The geometrical relationship between cameras and the vehicle body was identified by analyzing the physical structure of the fixtures through which cameras were mounted onto the Hexapod body.

Chapter 5

ALGORITHMS FOR EVEN TERRAIN LOCOMOTION

5.1 Introduction

As described in Chapters 3 and 4, the binocular vision system enables the Hexapod to perceive a specified foothold and the triangulation technique helps the vehicle to locate the foothold with respect to its current position. The next task the machine should do is to move in such a way that its right/left front foot steps on the designated foothold. This difficult task, which includes the control of each individual joint and the coordination of all 18 joints in real-time, is handled by sophisticated software algorithms inside a PDP-11/70 computer. Functionally speaking, the software may be divided into three levels. The lowest level, which deals with the servo control loop in generating commanded joint rate voltages by using a linear, error-driven scheme, is described in detail in [56]. The intermediate level, called the motion planning/execution level, is devoted to the computation of the desired position and velocity of each leg at each instant as seen from the center of the body. This involves the formulation of the kinematic equations which convert the Cartesian coordinates of a foot into the three corresponding joint angles and the reverse, the utilization of the inverse Jacobian matrix to obtain the desired joint rates from linear rates, the specification of the behavior of a supporting leg as viewed from vehicle body as the Hexapod is moving, and the generation of the trajectory of a foot in the transfer phase. A complete and detailed discussion of the above issues is given in [56]. The activities at the top level, namely the locomotion-cycle level, will be considered in this chapter. In Section 5.2, a particular gait suitable for FTL control is selected. The questions as to when, where, and how the body should move are addressed in Section 5.3. Particularly, a finite-step heuristic algorithm is devised to guide the center of body so that the stability margin is optimized. Section 5.4 explains the checking of the kinematic limit of every joint to prevent the Hexapod from getting into an undesired situation. Two approaches to updating the coordinates of the footholds as the Hexapod is moving are discussed in Section 5.5.

5.2 Selection of a Proper Gait

A gait is a sequence of events of leg lifting and leg placing. The problem of choosing a gait is to determine, within a locomotion cycle, which leg should be lifted up, how long it stays in the transfer phase before being placed down, and which leg should be lifted up next. With the gait defined, the legs in the supporting phase and in the transfer phase are known so that appropriate treatment is applied to each leg. For a six-legged vehicle, there are millions of possible gaits [4]. However, only a small number of them are suitable for FTL operation. In this mode, the middle legs have to step to the locations which were occupied by the front legs, and the rear legs must follow the footprints of the middle legs. Thus, in the same locomotion cycle, the front legs ought to move before the middle legs, and the rear legs should act only after the middle legs have completed the transfer phase; otherwise leg interference will occur. With the assumption that only one leg moves at a time, all gaits that satisfy the above requirement are shown in Fig. 5.1. Note that the legs of the Hexapod is labeled in the following way: leg 1 is the left, front leg; leg 2 is the right, front leg; leg 3 is the left, middle leg; leg 4 is the right, middle leg; leg 5 is the left, rear leg; and leg 6 is the right, rear leg. An inspection of the tree in this figure reveals that the right and left subtrees are mirror images of each other; i.e., by replacing '2' with '1', '4' with '3', and '6' with '5', the right subtree becomes identical to the left one. Due to this symmetry, it is sufficient to consider only one of the subtrees for further analysis. Among the 10 sequences in the left subtree, two are especially interesting. One is the sequence 1-3-5-2-4-6 and the other is 1-2-3-4-5-6. The former represents the case in which all three legs on the left side of the vehicle move before the motion switches to the right hand side. For the latter sequence, legs on each side move in an alternating way. The rest of the sequences are combinations of these two basic sequences. Furthermore, for purpose of this dissertation, sequence 1-3-5-2-4-6 is preferred over 1-2-3-4-5-6 because it has "physical symmetry", which is an important property and is commonly observed in animal motions [4]. A thorough study of this class of gaits (regular symmetric gaits) has been performed by Sun [39]. Therefore, 1-3-5-2-4-6 was chosen as the gait to realize FTL motion.

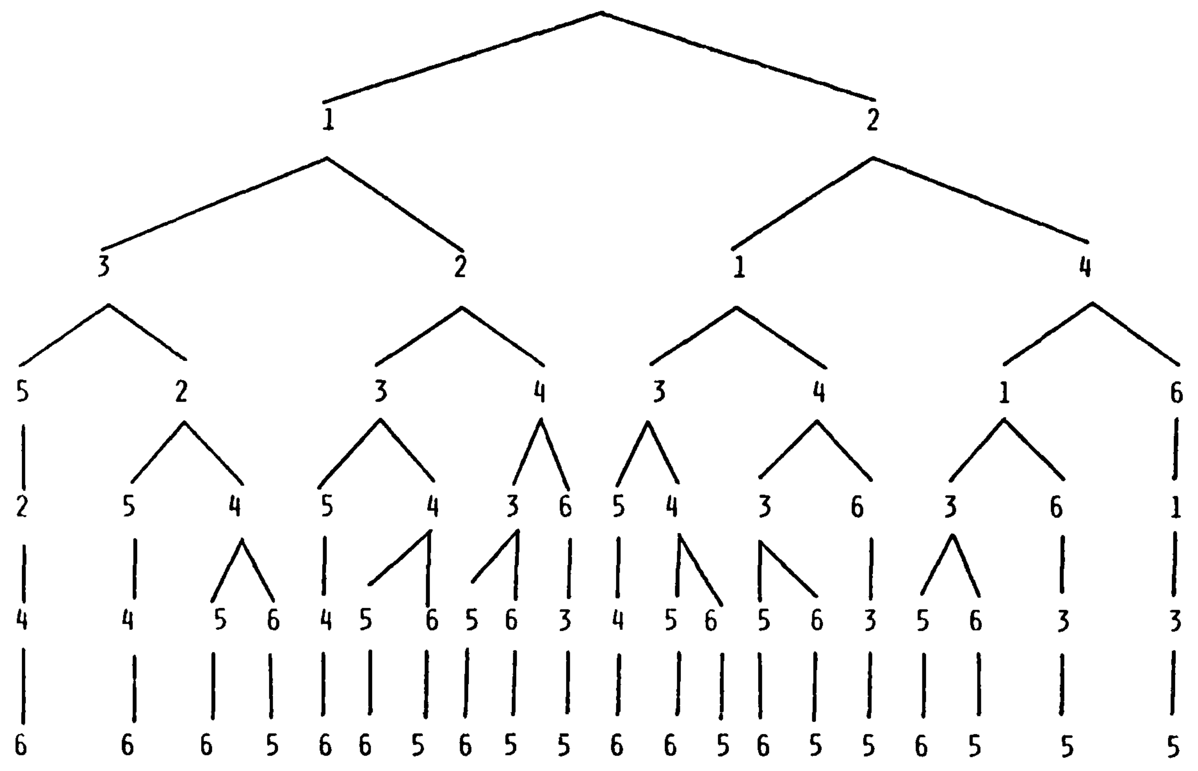


Figure 5.1 Possible Gait Sequences for Follow-The-Leader Operations.

5.3 Navigation of the Body

5.3.1 Timing--When Should the Body Move

With the sequence 1-3-5-2-4-6, a locomotion cycle begins at the moment when leg 1 is lifted up and ends when leg 6 is placed down. The timing which governs the motion of each leg is well defined. The question that remains to be answered concerns the motion of the body with respect to time; that is, whether it should move at a constant velocity throughout the whole cycle as in the case of wave gait locomotions [4], or whether the speed should be nonuniform and the body would only move for a portion of a cycle. Investigation of this issue requires the following definitions. Definitions 1 and 2 were given by McGhee in [4].

Definition 1: the supporting polygon of a mobile vehicle is a two-dimensional point set in a horizontal plane consisting of the convex hull of the vertical projection of all points of contact of each supporting foot.

Definition 2: the longitudinal stability margin of a supporting polygon is the shortest distance from the vertical projection of the center of gravity to an edge of this polygon as measured in the line of travel.

The timing problem mentioned above was evaluated in terms of the longitudinal stability margin, and the following example should provide some insight to this issue.

For simplicity, only straight-line motion will be considered and it will be assumed that only one leg is in the transfer phase at a time. Fig. 5.2.a shows the position of the center of body and the supporting polygon when leg 1 is moving forward. The front boundary of the polygon

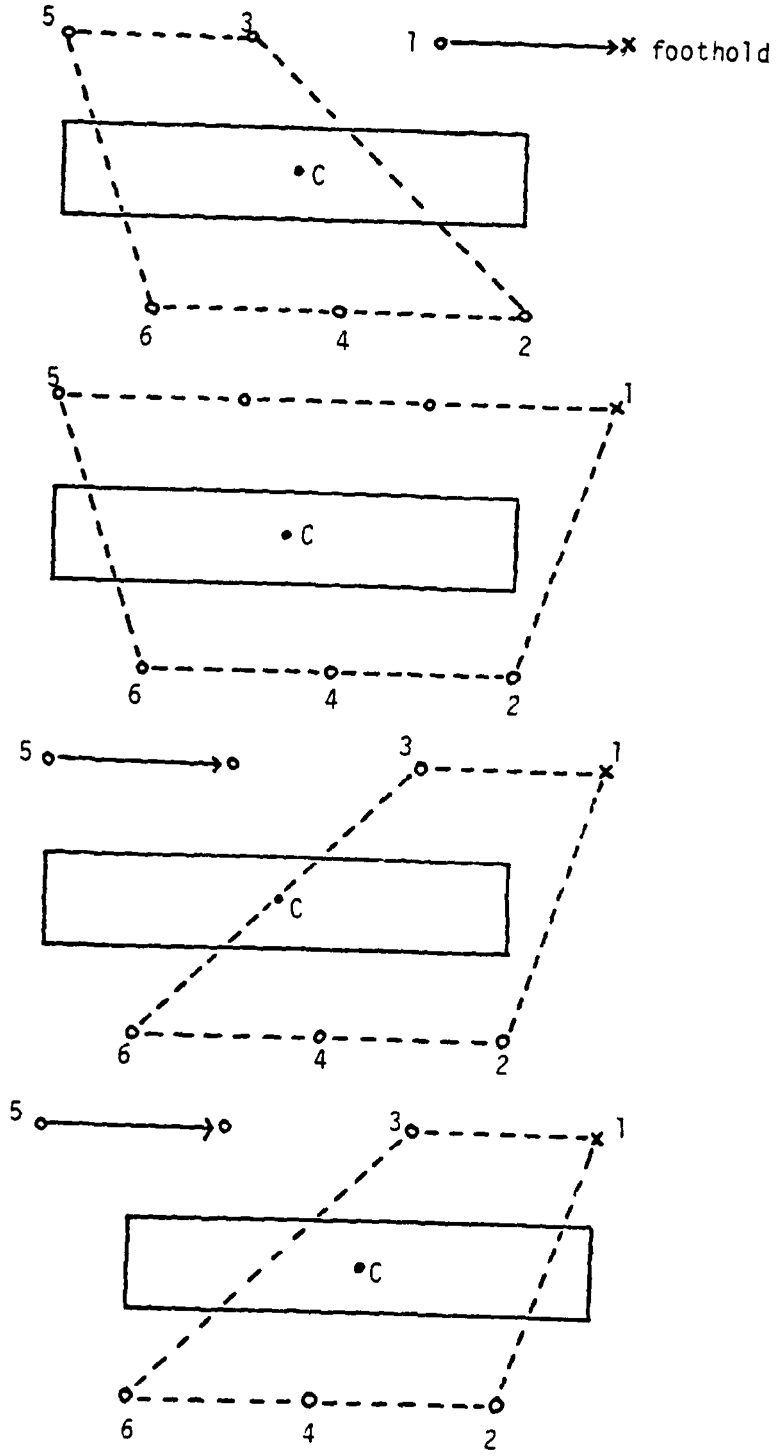


Figure 5.2 Supporting Polygon and the Location of the Center of the Body During a FTL Operation.

is defined by the foot tips of leg 2 and leg 3, which remain fixed before leg 1 is placed down. Clearly, if the body moves simultaneously with leg 1, its center would become closer and closer to line $\overline{32}$, which means that the longitudinal stability margin is strictly decreasing. Apparently, this is not a good choice. For the same reason, a similar outcome would be observed if the body tried to move while leg 2 is moving. Hence, it is concluded that the body should not move when leg 1 or leg 2 is in the transfer phase.

Fig. 5.2.b represents the case in which leg 3 is in the air. It is evident that, compared with the front legs or the rear legs, the middle legs have very little influence on the shape of the polygon as well as the stability margin, especially the longitudinal margin.

The next figure, Fig. 5.2.c shows the case in which the body has not moved since the beginning of the locomotion cycle and leg 5 is in motion. The center of the body is very close to the rear border $\overline{36}$ and the machine is in a dangerous position. This suggests that the vehicle body should move forward some distance before leg 5 takes off. Indeed, by doing so, the stabil'ty margin is greatly improved as seen in Fig. 5.2.d.

The timing diagram shown in Fig. 5.3 summarizes the foregoing discussion. It shows the intervals for leg movement, body motion and visual data acquisition. Events take place in a purely sequential fashion. Since all legs are in the supporting phase when the body is moving, this gait has the advantage of longitudinal stability margin in addition to programming simplicity. The drawback is that it is slow. The length of a cycle can be reduced by allowing two events to occur concurrently. Fig. 5.4 shows that body motion is overlapped with the

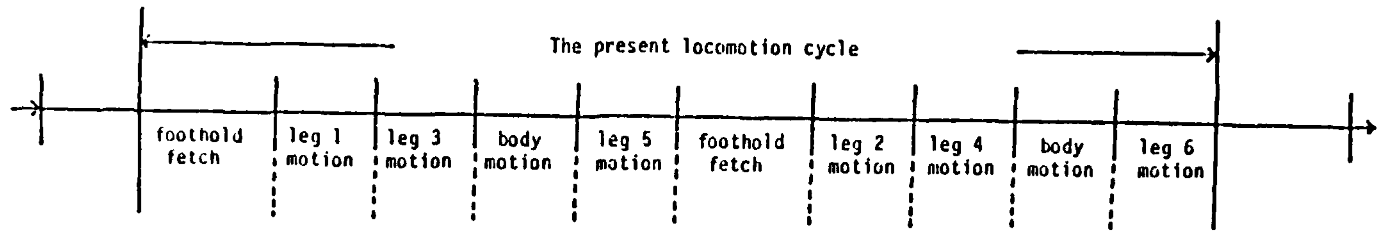


Figure 5.3 Timing Diagram of a Fully Sequential Motion.

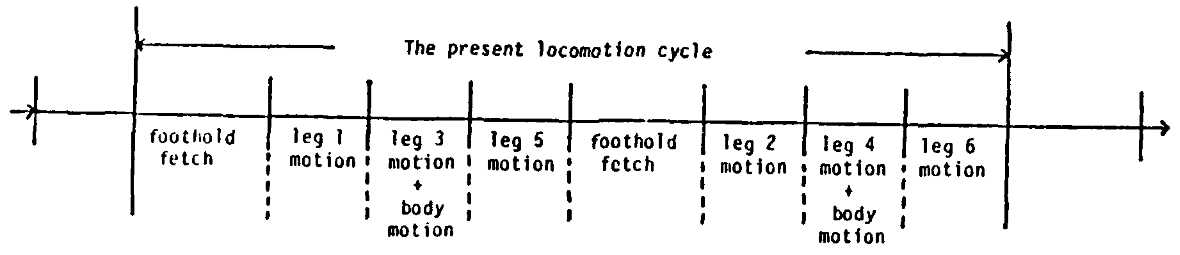

The present locomotion cycle
foothold fetch
leg 1 motion
leg 3 motion + body motion
leg 5 motion
foothold fetch
leg 2 motion
leg 4 motion + body motion
leg 6 motion

Figure 5.4 Timing Diagram Showing Overlapping of Body Motion and Leg Motion.

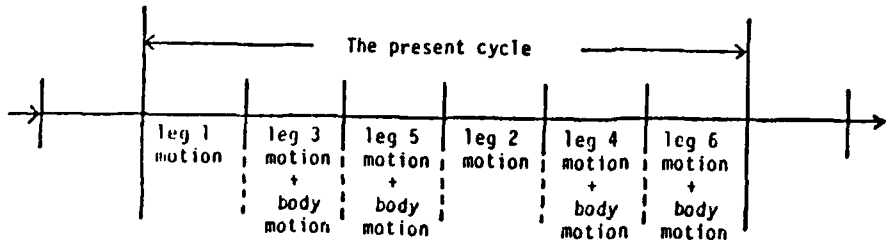

The present cycle
leg 1 motion
leg 3 motion + body motion
leg 5 motion + body motion
leg 2 motion
leg 4 motion + body motion
leg 6 motion + body motion

Figure 5.5 Timing Diagram with Overlapping of Body Motion, Leg Motion, and Foothold Fetch.

action of the middle legs. Foothold acquisition may also be overlapped with other events as shown in Fig. 5.5. This requires the multitasking of motion planning/execution module and vision software.

In terms of the software implementation, these three events were defined as LEGMOTION, BODYMOTION and FOOTHOLDFETCH. The flow of the events and the transitions from one to another are shown in the flowchart in Fig. 5.6. The activity represented by the flowchart is added to the existing motion planning/execution loop. Upon entering, the present event is identified and then the corresponding action follows. For a FOOTHOLDFETCH event, the vision subroutine is invoked and triangulation is performed to compute the spatial coordinates of the next foothold. The Y coordinate is used to distinguish footholds for leg 1 and leg 2. A variable, LEGLIFT, is defined to indicate the particular leg that is currently in the transfer phase. At the completion of FOOTHOLDFETCH, activity is switched to LEGMOTION. Each leg spends a fixed amount of time (determined by parameter LIFTTIME) in the transfer phase. If time has not expired yet, it continues to move toward the specified foothold. When the time is up, the next event has to be determined depending on the transferring leg. If it is either leg 1 or leg 2, the following event is still LEGMOTION, and LEGLIFT is updated as leg 3 or leg 4, respectively. If it is leg 3 or leg 4, BODYMOTION will be the next event. Otherwise, it must be either leg 5 or leg 6 and FOOTHOLDFETCH should follow. The vehicle body is allowed to move for a period of time specified by the parameter PERIOD. When this is done, the next event becomes LEGMOTION again and LEGLIFT is modified. The destination toward which the body is moving is discussed in the next section.

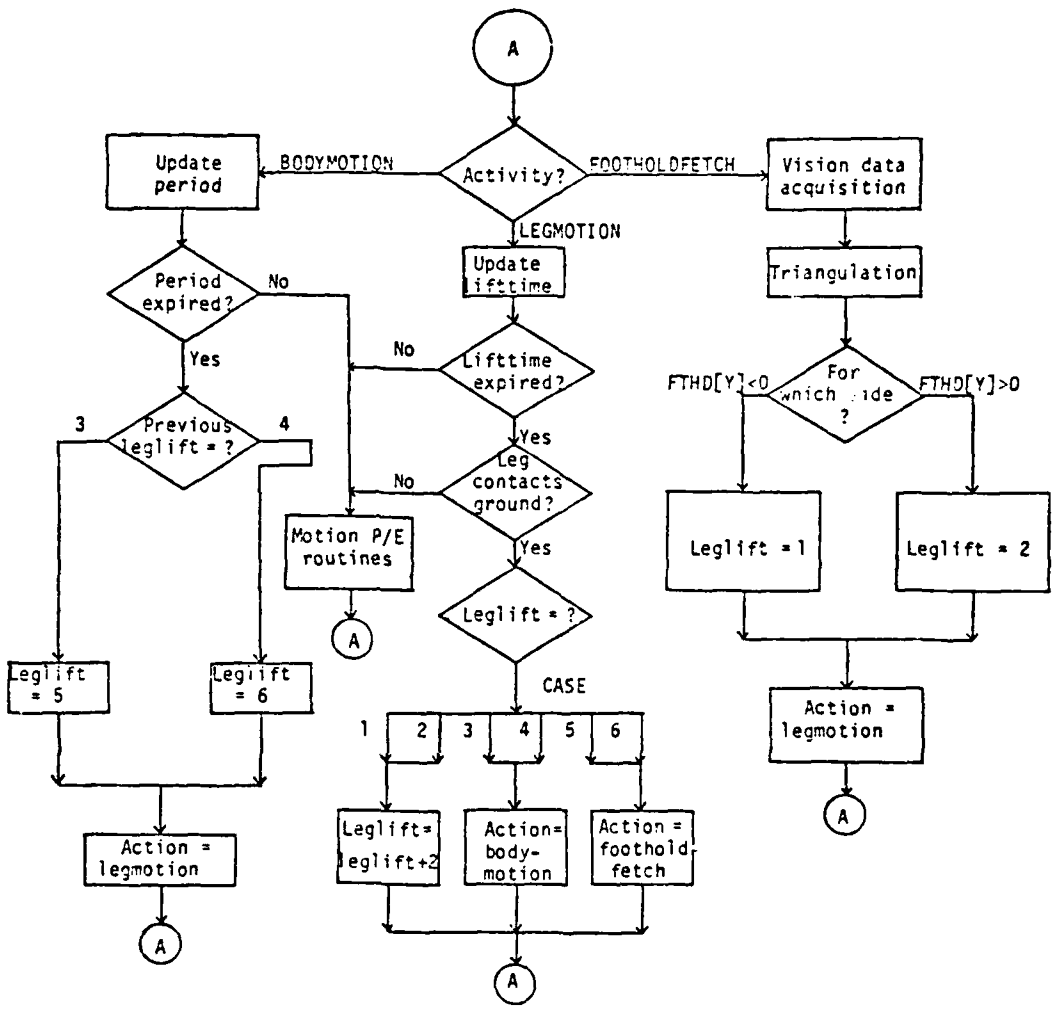


Figure 5.6 Flowchart of a Sequential FTL Motion.

5.3.2 Destination--Where Should the Body Move

For FTL operation, destination of each leg is its foothold (FTHD), which is specified explictly as follows:

$$\begin{aligned} &FTHD[I, K + 1] = FTHD[I-2, K] \quad \text{for } I = 3, 4, 5, 6 \\ &FTHD[I, K + 1] = \text{Assigned by operator for } I = 1, 2 \end{aligned} \qquad (5.1)$$

where I is the leg number and K+1 indicates the (K+1)th locomotion cycle. The next supporting polygon is actually known before the next cycle starts once the footholds are selected by the operator. This is illustrated in Fig. 5.7, where the present polygon is defined by points 1,2,3,4,5 and 6 and the next polygon is defined by points a,b,1,2,3 and 4.

While the destinations of legs are well defined from one cycle to another, the destination for the center of the body is less clear. If the current location of the center is at point C (Fig. 5.7), it is necessary to find a point C' within the next polygon as its destination. The following study attempts to find a point which would produce the maximal stability margin.

Before proceeding any further, an important distinction must be made here to distinguish the optimization procedure to be presented and the theory which has been proved for optimally stable wave gaits. In 1973, Umnov and Bessonov [57] proved that for forward, straight-line locomotion with a given duty factor (the duty factor, β_i, of leg i is the fraction of a locomotion cycle during which leg i is in contact with the supporting surface), there is a wave gait which gives an optimal longitudinal stability margin. An optimally stable wave gait has the

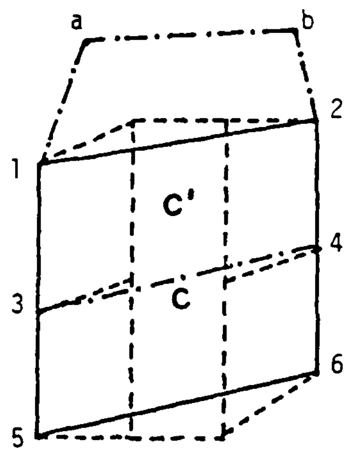


Figure 5.7 Present and the Next Supporting Polygons.

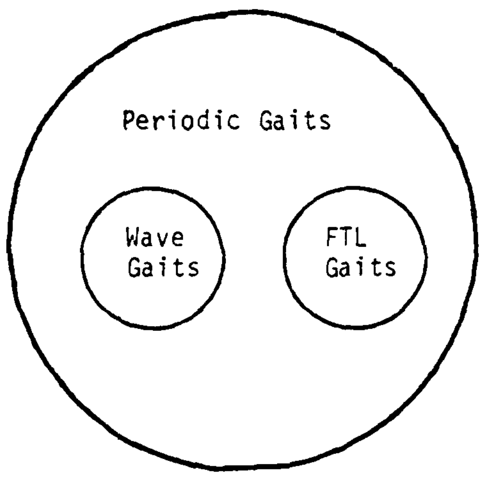


Figure 5.8 Relationships Between Periodic Gaits, Wave Gaits, and FTL Gaits.

following properties: (1) it is symmetric, and (2) the phase difference from front to back leg on each side is equal to 1-β (β is the duty factor for all legs), which implies that the rear legs move before the middle legs and the middle legs act before the front legs on both sides. According to this definition, the FTL gait (1-3-5-2-4-6) is not an optimally stable wave gait. Consequently, for any β, the stability of a FTL gait is less than that of a wave gait. In fact, FTL gaits and wave gaits are two different families of gaits whose intersection is empty, as illustrated in Fig. 5.8. While Umnov and Bessonov showed that the wave gait is optimal, among all possible gaits, for a special type of locomotion (forward, straight-line motion), the optimal algorithm developed in the next section attempts to find an optimal stability margin for any given gait with respect to any kind of motion. Thus, the optimal algorithm can be applied to FTL gaits as well as any other kind of gaits.

5.3.2.1 The Optimally Stable Point (OSP)

A few definitions are needed in order to identify the problem associated with finding of an optimal destination for the body.

Definition 3: The stability margin of a legged vehicle is the shortest distance from the vertical projection of the center of gravity of the body to each side of the supporting polygon.

Note that the stability margin is an extention of the longitudinal stability margin by considering not only the margin in the line of travel, but also margins in all other directions.

Definition 4: The Optimally Stable Point (OSP) of a supporting polygon is the interior point that produces the maximal stability margin when the center of gravity is projected onto it.

With these definitions, the problem to be solved may be stated as follows: given an arbitrary, convex, N-sided polygon, find its OSP.

Note that for Hexapod locomotions, N may be 3, 4, 5 or 6, while for a general N-legged walking machine, N may be greater than 6.

Three approaches were considered in computing the OSP of an arbitrary polygon. They are described below.

5.3.2.2 The Analytical Method

This approach first determines the equation of each side and then expresses the distance from an arbitrary point (x,y) inside the polygon to each side as a function of x and y. Generally, N distances, denoted as $[d_1,d_2,\ldots,d_n]$ would be obtained. The minimum of them is then maximized. A similar problem, which requires the minimization of the maximal joint angle of a multi-jointed manipulator, has been solved by Klein and Huang based on the ℓ-norm concept [58]. Although an exact solution requires that ℓ be infinity, they demonstrated that the result obtained with ℓ being 6 is close enough. While their problem was the minimization of a maximum quantity, the problem encountered here is to maximize a minimum value. However, the latter can be transformed into the former by substracting each variable from a constant. That is, to maximize the minimum of $[d_1,d_2,\ldots,d_n]$, these N quantities can be first transformed into $[c-d_1,c-d_2,\ldots,c-d_n]$. If di is the minimum of $[d_1,d_2,\ldots,d_n]$, then $c-d_i$ will be the maximal of $[c-d_1,c-d_2,\ldots,c-d_n]$. Thus, the minimization of $c-d_i$ implies the maximization of d_i. To minimize $c-d_i$, the ℓ-norm of $[c-d_1,c-d_2,\ldots,c-d_n]$ is used as a cost function, whose partial derivatives with respect to x and y are set to zero, respectively, to generate the equations needed for a minimal condition. If these equations can be solved analytically, the one can take the limit as ℓ goes to infinity to find the result. Otherwise, different values of ℓ may be substituted into the equations and numerical methods are then employed to compute the solutions. An example which illustrated the foregoing method is provide in [59].

As mentioned above, ℓ must be infinity to produce an exact solution, and there is no guarantee for a closed-form solution. However, if a smaller value of ℓ, say 6, can always yield a result that is close enough, this approach may be considered to compute the OSP for real-time control environments.

5.3.2.3 The Geometrical Method

This approach is best illustrated by examples. Consider the case of a triangle, which is the simplest polygon. It is evident that the OSP exists and is unique for all triangles. The OSP is located at the intersection of the three angular bisectors as shown in Fig. 5.9. Therefore the problem of finding the OSP is solved for any tripod-gait locomotion. Also note that the distances from this point to all sides are the same. This observation leads to the following theorem:

Theorem 1: If there exists one point O inside a polygon P such that the distance from O to each side is the same and is equal to S, then O is the optimally stable point and S is the maximal stability margin of P. Furthermore, O is located at the intersection of all the angular bisectors of P.

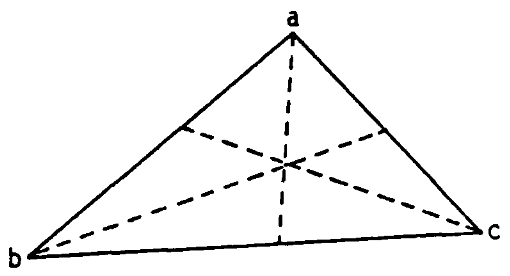


Figure 5.9 The Intersection of the Three Bisectors Gives the Optimally Stable Point.

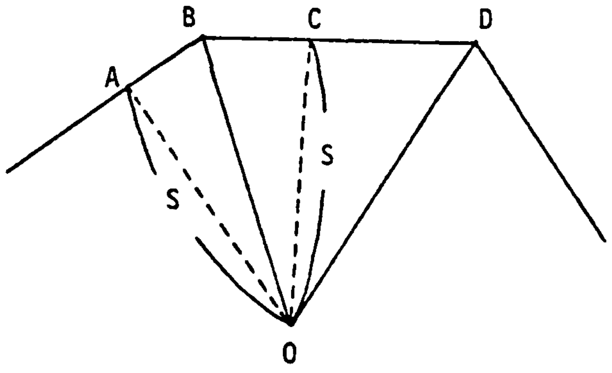


Figure 5.10 Portion of a Polygon Whose Angular Bisectors Intersect at a Single Point O.

Proof: refer to Fig. 5.10, which shows a portion of polygon P.

It is obvious that ΔABO is similar to ΔCBO, thus O lies along the angular bisector of angle ABC. Indeed, by connecting O to any vertex, it is apparent that O is a point along that bisector. Therefore, O is located at the intersection of all bisectors.

Since O is the intersection of all bisectors, by connecting O to all vertices gives N triangles, and O is the common vertex of all triangles. If point O is moved away from its location, it would be inside a triangle, say ΔBOD. It is evident that the distance from O to side BD is now less than S. Hence, O must be the OSP. Q.E.D.

Definition 5: A polygon whose angular bisectors intersect at unique point is called a "BI polygon". The set of all BI polygons is denoted as BIP.

Definition 6: A polygon is regular if and only if (1) all its sides have the same length, and (2) all its internal angles are equal.

From Theorem 1, it is evident that the geometrical method may be applied to only the set of BIP, which is only a small portion of the set of all polygons. Note that the set of regular polygons is a subset of BIP. One interesting property of a regular polygon is that the OSP is also its geometric center, which is defined as follows:

Definition 7: The coordinates (X_{gc}, Y_{gc}) of the Geometric Center (GC) of a N-sided polygon is defined as follows:

$$X_{gc} = (X_1+X_2+\ldots+X_n)/N$$
$$Y_{gc} = (Y_1+Y_2+\ldots+Y_n)/N \qquad (5.2)$$

where X_i and Y_i are the coordinates of the ith vertex.

Theorem 2: The angular bisectors of a regular polygon P intersect at a single point O, which is also the geometric center of P.

Proof: assume that P has N sides, its N vertices are labeled as V_0 through V_{n-1}, and the coordinate system is defined in such a way that O is the origin and the X axis passes V_0. It is also assumed that the distance from O to each vertex has been normalized to be 1. This arrangement is shown in Fig. 5.11. It is evident that the Cartesian coordinates of V_k are $(\cos(k\theta), \sin(k\theta))$ where θ is $2\pi/N$, thus

$$X_{gc} = \frac{1}{N}[\cos(0) + \cos(\theta) + \cos(2\theta) + \ldots + \cos(n-1)\theta] \qquad (5.3)$$

$$Y_{gc} = \frac{1}{N}[\sin(0) + \sin(\theta) + \sin(2\theta) + \ldots + \sin(n-1)\theta] \qquad (5.4)$$

Eqs. (5.3) and (5.4) resemble the following trigonometric equations

$$\cos\phi + \cos(\phi + \alpha) + \cos(\phi + 2\alpha) + \ldots + \cos(\phi + (n-1)\alpha)$$
$$= \frac{\sin\frac{n\alpha}{2}}{\sin\frac{\alpha}{2}} \cdot \cos\left(\phi + \frac{n-1}{2}\alpha\right) \qquad (5.5)$$

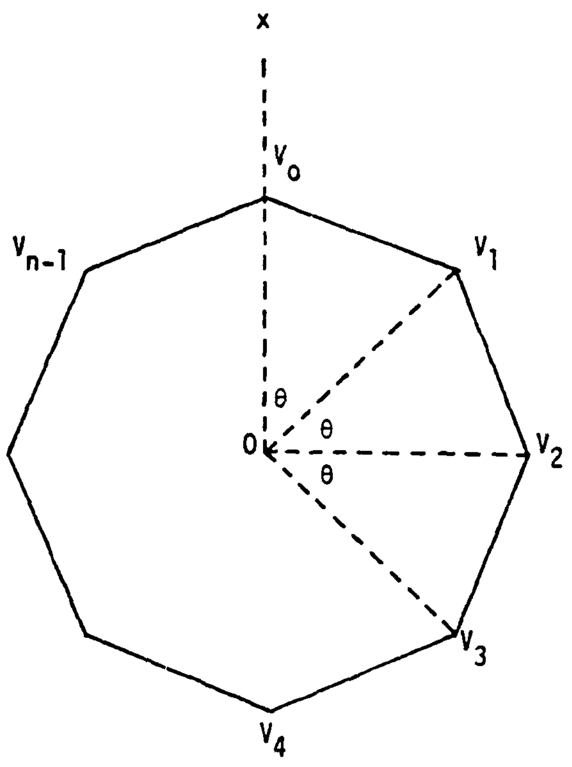


Figure 5.11 A Regular Polygon with N Sides.

$\sin\theta + \sin(\phi + \alpha) + \sin(\phi + 2\alpha) + \ldots. + \sin(\phi + (n-1)\alpha)$

$$= \frac{\sin\frac{n\alpha}{2}}{\sin\frac{\alpha}{2}} \cdot \sin\left(\phi + \frac{n-1}{2}\alpha\right) \qquad (5.6)$$

Comparing them reveals that

$$X_{gc} = \frac{\sin\pi}{\sin\frac{\pi}{N}} \cdot \cos\left(\frac{n-1}{n}\pi\right) = 0 \qquad (5.7)$$

$$Y_{gc} = \frac{\sin\pi}{\sin\frac{\pi}{N}} \cdot \sin\left(\frac{n-1}{n}\pi\right) = 0 \qquad (5.8)$$

Therefore O is also the GC of P. Q.E.D.

To show that this method can not be generalized to handle an arbitrary polygon, a counter-example is provided in Fig. 5.12, where the polygon is a rectangle. It is evident that in this case the four angular bisectors do not intersect at one point. Moreover, the OSP is not unique since every point along the segment AB will yield the same stability margin which is with respective to L2 and L4. This example brings out two points: (a) the bisectors of an arbitrary polygon generally do not intersect at one point, and (b) the OSP may not be unique.

Since the geometrical approach of finding the intersection point can only be applied to BIP, a more general method is needed to deal with an arbitrary polygon. This motivation leads to the development of the following algorithm.

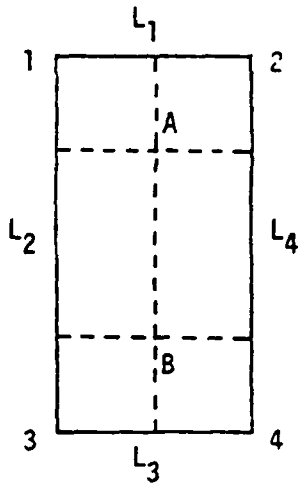


Figure 5.12 A Rectangle with More Than One OSP.

5.3.2.4 A Heuristic Algorithm

More definitions are needed in order to describe this algorithm.

Definition 8: Two polygons P_1 and P_2 are similar to each other if and only if (1) every pair of corresponding angles are equal, and (2) the ratio of the length of the corresponding sides is a constant, i.e.,

$$\frac{|\bar{V}_1{}'|}{|\bar{V}_1|} = \frac{|\bar{V}_2{}'|}{|\bar{V}_2|} = \frac{|\bar{V}_3{}'|}{|\bar{V}_3|} = \dots = \frac{|\bar{V}_n{}'|}{|\bar{V}_n|} = r \qquad (5.9)$$

where $\bar{V}i$ and $\bar{V}i'$ are side i of P1 and P2, respectively. The similarity between P1 and P2 is denoted as P2=r(P1).

Definition 9: To "shrink" a polygon P by D, a line is drawn in parallel to each individual side and is moved D inward from the original side, as shown in Fig. 5.13. The shrunken polygon is denoted as SP.

Definition 10: SP is degenerated if P is an N-sided polygon while SP has less than N sides.

To fulfill this algorithm, the following procedures should be followed after an initial polygon (IP) is given:

(1) let K = 1 and P(1) = IP, where the index K indicates the number of iterations.

(2) Compute the geometric center of P(K).

(3) Compute the distances from GC (Geometric Center) to each side of P(K). N distances, $(d_1,d_2,\dots,d_n)$, would be obtained. Let the minimal of them be D_k, i.e., $D_k = \mathrm{MIN}(d_1,d_2,\dots,d_n)$.

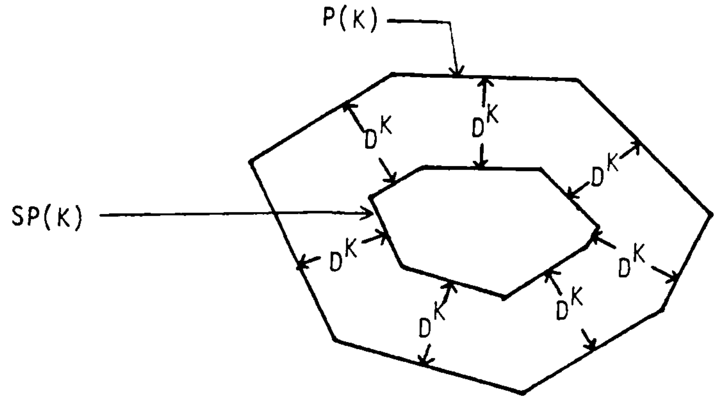


Figure 5.13 Shrinking a polygon by D.

(4) "Shrink" P(K) by D_k and denote the shrunken polygon as SP(K). Note that SP(K) may be (i) "similar" to P(K), (ii) degenerated, or (iii) neither (i) nor (ii).

(5) Examine SP(K) to see if it can be further shrunk. Since the most simple type of polygon is a triangle, SP(K) has to be at least a triangle or have more than three sides to continue the shrinking process. If SP(K) can be processed further, let K = K+1, P(K) = SP(K-1) and repeat steps (2) through (4). Otherwise, SP(K) may be a point or a segment, then go to step (6) and the process is terminated.

(6) The optimal stability margin is

$$D_{osm} = \sum_{k=1}^{N} D_k \qquad (5.10)$$

The following discussion is intended to show that the above algorithm indeed generates an optimal stability margin. The analysis is divided into two parts, one deals with the cases where SP(1), the first shrunken polygon, is similar to IP, and the other handles situations in which SP(1) is NOT similar to IP.

5.3.2.2 Cases Where SP(1) is Similar to IP

Theorem 3: SP(1) is similar to IP if and only if the term

$$\frac{\cot(\frac{\theta_i}{2}) + \cot(\frac{\theta_{i+1}}{2})}{|\bar{V}_i|}$$

is a constant for every side of IP. The parameter $|\bar{V}i|$ is the length of side i, and θ_i and $\theta_i + 1$ are the two angles associated with it.

Proof: Let $\bar{V}_i$ and $\bar{V}_i'$ be the ith side of IP and SP(1). Refer to Fig. 5.14, which shows portion of IP and SP(1) with $\bar{V}_i$ and $\bar{V}_i'$ both present. If IP is shrunk by D to produce SP(1), it is clear that

$$\begin{aligned} |\bar{V}_i| &= |\bar{V}_i'| + a + b \\ &= |\bar{V}_i'| + D \cdot \cot(\tfrac{\theta i}{2}) + D \cdot \cot(\tfrac{\theta i+1}{2}) \\ &= |\bar{V}_i'| + D \cdot [\cot(\tfrac{\theta i}{2}) + \cot(\tfrac{\theta i+1}{2})] \qquad (5.11) \end{aligned}$$

Rearranging Eq. (5.11) gives

$$|\bar{V}_i'| = |\bar{V}_i| - D \cdot [\cot(\tfrac{\theta i}{2}) + \cot(\tfrac{\theta i+1}{2})] \qquad (5.12)$$

Hence the ratio between $|\bar{V}_i'|$ and $|\bar{V}_i|$ is

$$\begin{aligned} \frac{|\bar{V}_i'|}{|\bar{V}_i|} &= \frac{|\bar{V}_i'| - D \cdot [\cot(\tfrac{\theta i}{2}) + \cot(\tfrac{\theta i+1}{2})]}{|\bar{V}_i|} \\ &= 1 - D\,[\cot(\tfrac{\theta i}{2}) + \cot(\tfrac{\theta i+1}{2})]/\,|\bar{V}i| \qquad (5.13) \end{aligned}$$

Since D is the same for each pair of sides, the ratio for any pair of sides would be the same if the term

$$\frac{\cot(\tfrac{\theta i}{2}) + \cot(\tfrac{\theta i+1}{2})}{|\bar{V}i|}$$

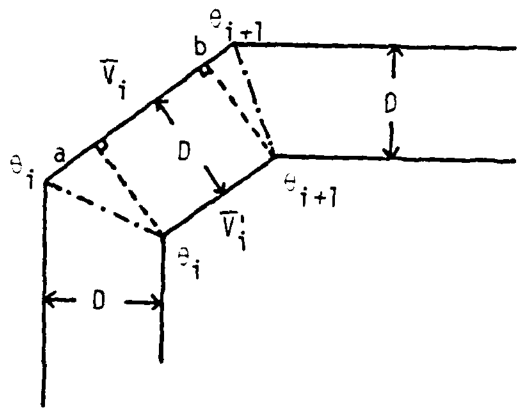


Figure 5.14 Relationship Between Two Corresponding Sides of Two Similar Polygons.

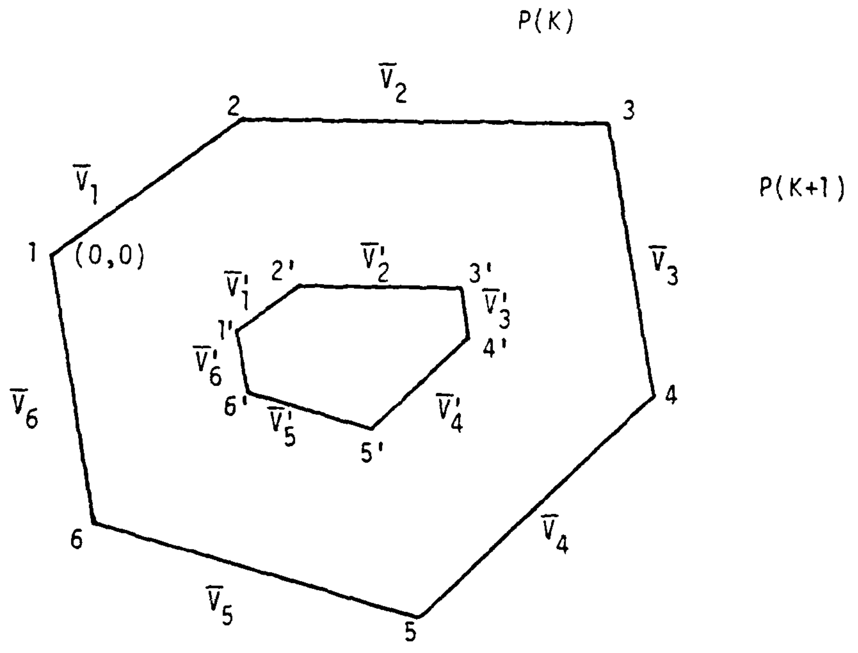


Figure 5.15 The Coordinate Systems Associated with the Original and the Shrunken Polygons.

is a constant. If this the case, then SP(1) is similar to IP by definition 6. Q.E.D.

Theorem 4: If $P_2 = r(P_1)$, $[d_1,d_2,\ldots,d_n]$ is the distances from the GC of P_1 to its sides, and $[d_1',d_2',\ldots,d_n']$ is those of P_2, then

$$\frac{d_1'}{d_1} = \frac{d_2'}{d_1} = \ldots = \frac{d_n'}{d_n} = r \tag{5.14}$$

Proof: One may assign a coordinate system to P_1 and P_2, respectively. The origin of this coordinate system is at vertex 1, and each side is represented as a vector, as shown in Fig. 5.15. Following this notation, the Cartesian coordinates of the vertices of P_1 are:

$$1(x,y) = (0,0),$$

$$2(x,y) = \overline{V_1},$$

$$3(x,y) = \overline{V_1} + \overline{V_2},$$

$$n(x,y) = \overline{V_1} + \overline{V_2} + \ldots + \overline{V_{n-1}}. \tag{5.15}$$

and those of P_2 are:

$$1'(x,y) = (0,0),$$

$$2'(x,y) = \overline{V_1}',$$

$$3'(x,y) = \overline{V_1}' + \overline{V_2}',$$

$$n'(x,y) = \overline{V_1}' + \overline{V_2}' + \ldots + \overline{V_{n-1}}'. \tag{5.16}$$

Thus the coordinates of the GC of P_1 are

$$GC(x,y) = [\overline{0} + \overline{V_1} + (\overline{V_1} + \overline{V_2}) + \ldots + (\overline{V_1} + \overline{V_2} + \ldots + \overline{V_{n-1}})]/N$$

$$= [(n-1)\overline{V_1} + (n-2)\overline{V_2} + \ldots + \overline{V_{n-1}}]/N$$

$$= (X_{gc}, Y_{gc}) \qquad (5.17)$$

and those of P_2 are

$$GC'(x,y) = [(n-1)\overline{V_1}' + (n-2)\overline{V_2}' + \ldots + \overline{V_{n-1}}']/N$$

$$= (X_{gc}', Y_{gc}') \qquad (5.18)$$

Since $P_2 = r(P_1)$, thus

$$\frac{|\bar{V_i}'|}{|\bar{V_i}\ |} = \gamma \qquad (5.19)$$

Substituting Eq. (5.19) into (5.18) and comparing the result with (5.17) reveals that

$$X_{gc}' = r \cdot X_{gc}$$

$$Y_{gc}' = r \cdot Y_{gc} \qquad (5.20)$$

Let the equation of V_1 in P_1 coordinate system be

$$a\ X + b\ Y = c \qquad (5.21)$$

Since this line passes through the origin, thus c must be zero and the equation becomes

$$a\ X + b\ Y = 0 \qquad (5.22)$$

Since $\bar{V}_1'$ is parallel to $\bar{V}_1$, the equation of $\bar{V}_1'$ in P_2 coordinate system is also $a X + b Y = 0$. The distance from a point (x,y) to a line $aX + bY = c$ is

$$\frac{|ax + by - c|}{\sqrt{a^2 + b^2}} \tag{5.23}$$

Hence

$$d_1 = \frac{|a\,Xgc + b\,Ygc|}{\sqrt{a^2 + b^2}} \tag{5.24}$$

$$d_1' = \frac{|a\,Xgc' + b\,Ygc'|}{\sqrt{a^2 + b^2}} = \frac{r\,|a\,Xgc + b\,Ygc|}{\sqrt{a^2 + b^2}} \tag{5.25}$$

Therefore

$$d_1' = r\,d_1 \tag{5.26}$$

By applying the same technique to each vertex and each side

Theorem 4 is proved. Q.E.D.

<u>Corollary 1</u>: if $P_2=r(P_1)$, $D=MIN[d_1,d_2,\ldots,d_n]$, and

$D'=MIN[d_1',d_2',\ldots,d_n']$, then

$$\frac{D'}{D} = \frac{|\bar{V}_i'|}{|\bar{V}_i|} = r \tag{5.27}$$

for $i = 1$ to N.

Theorem 5: if SP(1)=r(IP), then SP(K)=r(P(K)) and $D_{k+1} = rD_k$.

Proof: This statement can be proved by induction as follows:

(i) the case of N = 1 holds by the given assumption SP(1)=r(IP) and by Corollary 1.

(ii) Suppose that it is true for N = K, i.e., SP(K)=P(K+1)=r(P(K)), what remains to be verified is that it is also true for N = K+1, i.e, SP(K+1)=P(K+2)=r(P(K+1)). Let $\bar{V}_i$, $\bar{V}_i'$, and $\bar{V}_i''$ be the corresponding side of P(K), P(K+1), and P(K+2), and that P(K) is shrunk by D_k to produce P(K+1) while P(K+1) is shrunk by D_{k+} 1 to produce P(K+2). From Fig. 5.14 and Eq. (5.13), the following equations hold:

$$\frac{|\bar{V}_i'|}{|\bar{V}_i|} = 1 - \frac{D_k \cot(\frac{\alpha i}{2}) + \cot(\frac{\theta i+1}{2})}{|\bar{V}i|} \qquad (5.28)$$

$$\frac{|\bar{V}_i''|}{|\bar{V}_i'|} = 1 - \frac{D_{k+1} \cot(\frac{\theta i}{2}) + \cot(\frac{\theta i+1}{2})}{|\bar{V}i'|} \qquad (5.29)$$

Since P(K+1) = rP(K), by Corollary 1,

$$\frac{D_{k+1}}{|\bar{V}i'|} = \frac{D_k}{|\bar{V}i|} \qquad (5.30)$$

Substituting Eq. (5.30) into (5.28) and (5.29) yields

$$\frac{|\bar{V}i''|}{|\bar{V}i'|} = \frac{|\bar{V}i'|}{|\bar{V}i|} = \gamma \qquad (5.31)$$

Hence, by the definition of similarity, P(K+2)=r(P(K+1)), and

$$D_{k+1} = \gamma \cdot D_k \text{ from Corollary 1.}$$

(iii) the statement holds for any value of K by induction. Q.E.D.

Theorem 6: if SP(1)=r(IP), then IP will be shrunk into a single point after the shrinking process has been applied an infinite number of times, and the optimal stability margin is

$$D_{osm} = \sum_{k=1}^{\infty} D_k = D_1 + rD_1 + r^2 D_1 + r^3 D_1 + \dots$$

$$= D_1/(1-r) \qquad (5.32)$$

Proof: According to Theorem 5, if SP(1)=r(IP), then every time the shrinking operation is applied to a polygon P(K), SP(K) is still similar to P(K) but is reduced by a factor of r. Mathematically, it takes an infinite number of times of shrinking to reduce IP into a point O. Thus Eq. (5.10) becomes Eq. (5.32). Q.E.D.

Theorem 7: if SP(1)=r(IP), then IP is a member of BIP and the OSP is the intersection point of all bisectors.

Proof: By Theorem 6, if SP(1) = r(IP), then IP will eventually be shrunk into a single point O. Since every side of IP approaches to O by the same amount of distance each time the shrinking operation is applied, the distances from O to all sides are equal and have a value of D_{osm}. Thus the angular bisectors of IP intersect at one point and it is a BI polygon. Q.E.D.

Theorem 8: if IP is a BI polygon, then SP(1)=r(IP).

Proof: Since IP is a BI polygon, all its bisectors intersect at a single point O and the distances from O to each side are all equal.

Refer to Fig. 5.16, which shows point 0 and the ith side of IP. Evidently,

$$|\bar{V}_i| = a + b = S\cot(\frac{\theta_i}{2}) + S\cot(\frac{\theta_{i+1}}{2}) \qquad (5.33)$$

where S is the distance from 0 to side i. Substituting $|\bar{V}i|$ into the term of theorem 3 reveals that it gives a constant 1/S. Moreover, since S is the same for every side of IP, hence the condition stated in Theorem 3 is true and SP(1) is similar to IP. Q.E.D.

<u>Theorem 9</u>: if SIP is the set of all IP which have the property of SP(1)=r(IP), then SIP = BIP.

<u>Proof</u>: From Theorem 7, one knows that if IP∈SIP then IP∈BIP, which implies that SIP⊂BIP. On the other hand, from Theorem 8, if IP∈BIP then IP∈SIP, thus BIP⊂SIP. Therefore SIP = BIP. Q.E.D.

Although BIP polygons are of theoretical interest and their OSP can be easily computed, they are not likely to occur in rough terrain locomotion. Thus, from practical point of view, polygons which are not BIP are more closely related to vehicle locomotion. The following analysis deals with polygons of this class.

5.3.2.3 Cases where SP(1) Is Not Similar To IP

For these cases, the relation

$$D_{osm} = \sum_{k=1}^{N} D_k$$

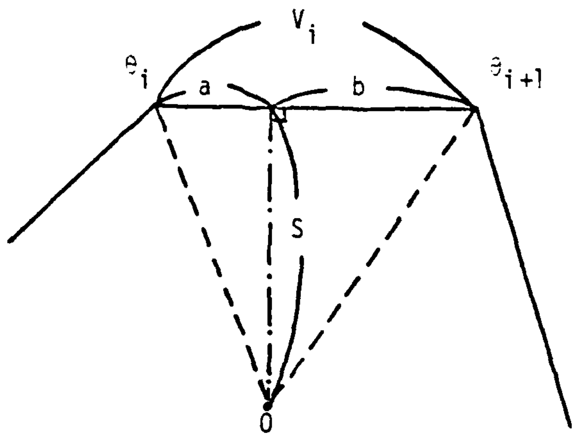


Figure 5.16 The ith Side of a BI polygon.

still holds, however, the ratio between D_{k+1} and D_k is no longer a constant, hence the summation can not be expressed in the same closed form as in Eq. (5.16).

D_{osm} is optimal by the principle of optimality [60], which states that, in a multistage path, the optimal path is comprised by each optimal subpath between two neighboring stages. To relate the principle of optimality to the optimal algorithm described earlier, one may represent the algorithm as a multistage path, where the starting point is the IP and SP(1), SP(2), ... are the successive nodes along the path. Moreover, there are only two paths between neighboring nodes, one will lead to the OSP while the other will not. When applying the algorithm, it is clear that the region outside the shrunken polygon will not give the optimal result, thus it is excluded from further consideration, i.e., only the SP is investigated in the next iteration. This point is illustrated by the example shown in Fig. 5.17 where the IP is a trapezoid. After finding the GC and shrinking it once, SP(1) degenerates into two triangles. It is apparent that Δcde needs not be considered any further since every point inside it gives a distance to sides L2 and L3 that is shorter than D. In other words, all points inside the trapezoid except Δabc are excluded from further consideration, and the OSP of Δabc is also the OSP of the initial trapezoid.

For practical purpose, the shrinking process may be terminated if D is smaller than a threshold, and the algorithm can be considered as finite-step in this sense. At least, it can be stated that the algorithm is "D_k near optimal" or "optimal within D_k". The threshold may be set according to the accuracy of the vehicle. For instance, the posi-

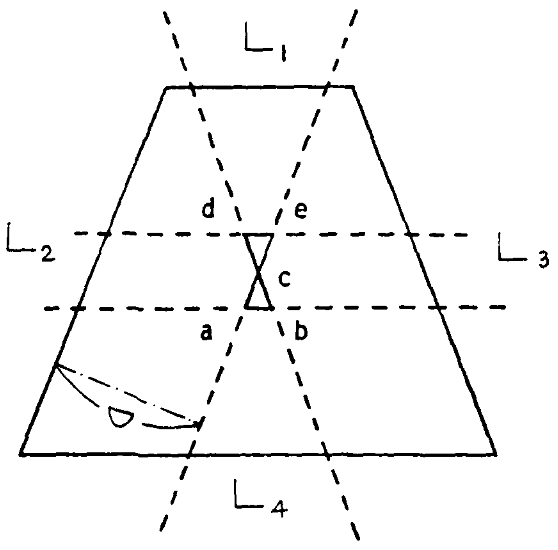


Figure 5.17 A Trapezoid witha Unique OSP.

tion control of the OSU Hexapod has an accuracy of 0.5 inch, and this value may be selected as the threshold.

5.3.2.4 Classification of Polygons

Polygons may be divided into the following three catagories based on the nature of OSP and the maximum stability margin.

(1) Polygons whose OSP is unique and the distances from the OSP to all side are the same. The set of these polygons is described earlier as BIP or SIP. Fig. 5.18 shows a polygon which belongs to this group but is not regular. This polygon is created as follows. First, the OSP O is specified. Then the four segments OA, OB, OC, and OD, which have the same length, are drawn in such a way that the angles between them (angles 1, 2, 3, and 4) are not all equal. Lines which are perpendicular to one of the four segments are then created, and the intersections of these lines define the resultant polygon.

(2) Polygons whose OSP is not unique. An example is the rectangular given in Fig. 5.12. As mentioned before, all points within segment AB provide the same optimal stability margin with respect to sides L2 and L4.

(3) Polygons whose OSP is unique but the distances from the OSP to all sides are not equal. The trapezoid shown in Fig. 5.17 is such an example. The OSP of this polygon is also the OSP of Δabc, and it is obvious that the distances from the OSP to the four sides are not the same.

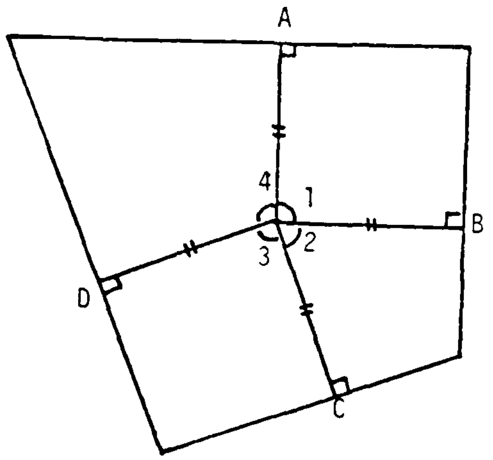


Figure 5.18 A Non-Regular BI Polygon.

5.3.2.5 Remarks about the Algorithm

A few remarks related to the optimal algorithm are given below.

(1) Two assumptions were made for the development of the algorithm. First, the footholds of all six legs were assumed to be known before the body moved. Second, in defining the supporting polygon and the projection of the center of the body, it was assumed that all legs were massless.

(2) No assumption was made on the type of gait nor the type of locomotion. Thus it is applicable to any kind of gait for any type of motion.

(3) Since the execution of the algorithm is rather time-consuming, the geometric center is used as the destination of the center of the body in the current software.

5.3.2.6 Utilization of the OSP in FTL Motion

The OSP concept is incorporated into the current software to realize the FTL operation in the following way. First, the GC of the present, 6-vertex supporting polygon, denoted as OSP1, is computed. After a foothold is assigned for leg 1, the next, 6-vertex supporting polygon, which is formed with leg 1, 3, and 5 at their new positions while legs 2, 4, and 6 in the old positions (Fig. 5.19), is projected and its GC, denoted as OSP2, is computed. Then during the period of body motion, the center of the body moves from OSP1 toward OSP2, which occurs after middle leg placement and before rear leg placement. During the second half of the locomotion cycle, a foothold is assigned for leg 2. Again, the following, 6-vertex supporting polygon with all 6 legs in

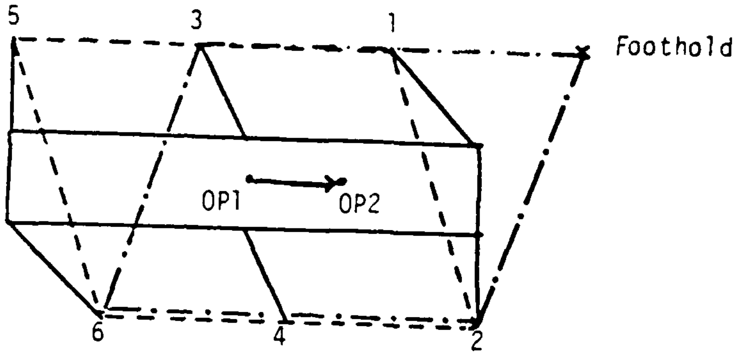


Figure 5.19 The Polygons Used to Determine the Motion of the Body for the First Half Cycle.

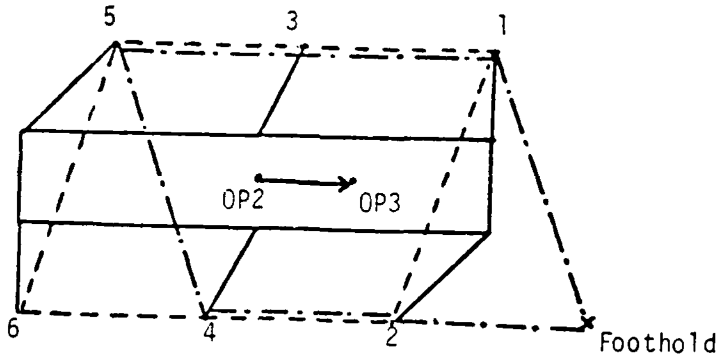


Figure 5.20 The Polygons Used to Determine the Motion of the Body for the Second Half Cycle.

their new locations (Fig. 5.20) is calculated and its GC, denoted as OSP3, is computed. Then the body will move from OSP2 toward OSP3. Note that only three, 6-vertex polygons have been taken into consideration in the above discussion. However, in an actual FTL locomotion, more than three supporting polygons are involved. Furthermore, some of them are 5-vertex polygons and the others are 6-vertex. All the polygons which show up, one time or another, during an entire FTL motion are given in Fig. 5.21. There are totally 9 different supporting polygons for a complete locomotion cycle provided that only one leg moves at a time. Fig. 5.21.a is the initial configuration which shows both the Hexapod body and the 6-vertex polygon. The numbers indicate the actual leg position at that time. In Fig. 5.21.b, leg 1 is moving toward foothold 1 thus the polygon is a 5-vertex one. Whenever both leg 1 and leg 5 are in the supporting phase, the shape of the polygon does not change regardless if leg 3 is in the air or on the ground. This situation is reflected in Fig. 5.21.c, where the missing number "3" implies the "don't care" condition for leg 3. When leg 5 is moving, another 5-vertex polygon appears as shown in Fig. 5.21.d. The rest of these figures can be interpreted in a similar way. Comparing Figs. 5.19, 5.20, and 5.21, it is clear that the present software only considers 3 of the 9 polygons, which are (a), (e), and (i). In order to faithfully fulfill the concept of OSP, all polygons should be considered, their OSPs should be computed, and the body should react in such a way that whenever a new polygon appears, it would move toward the OSP of the newest polygon. By this philosophy, not only that the body has to move more frequently, as opposed to the adopted scheme where the body motion is confined to two specific instants, but also that the vehicle has to

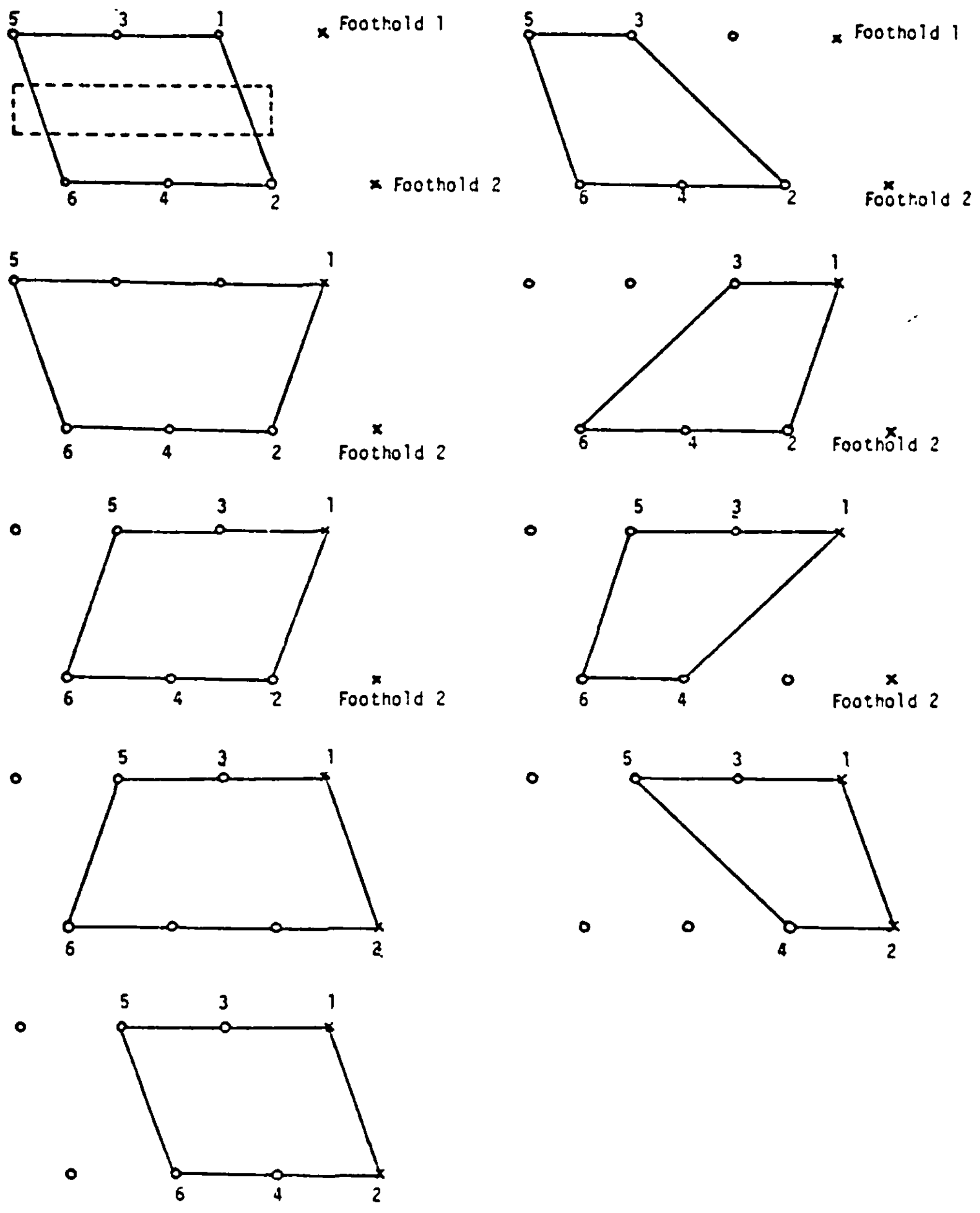


Figure 5.21 All Polygons Associated with a FTL Locomotion.

move back and forth to accommodate the drifting of OSPs, which may not be desirable because of the backward motion involved. To illustrate, the geometric center, the front longitudinal margin, the rear longitudinal margin, and the resultant longitudinal stability margin of each of the 9 polygons are calculated and listed in Table 5.1. The GC is used in place of the OSP for the simplicity of computation. These quantities are calculated as follows. First, the initial position of each leg and the two footholds are assigned as

$$
\begin{aligned}
\text{leg } 1 &= (0.3L, -K), \\
\text{leg } 2 &= (0.5L, K), \\
\text{leg } 3 &= (-0.1L, -K), \\
\text{leg } 4 &= (0.1L, K), \\
\text{leg } 5 &= (-0.5L, -K), \\
\text{leg } 6 &= (-0.3L, K), \\
\text{FTHD } 1 &= (0.7L, -K), \\
\text{FTHD } 2 &= (0.9L, K),
\end{aligned}
\tag{5.34}
$$

where L is the body length of the vehicle and is measured as the distance between a front and a rear legs, K is half of the body width, and the origin is the center of the body. These positions are assigned in this way so that the distance between the present position and the foothold of all legs are equal. The quantities in Table 5.1 then can be computed by using the coordinates of those vertices which define the polygon. For example, consider Fig. 5.21.a. This polygon is defined by legs 1, 2, 5, and 6, thus its GC is

TABLE 5.1

The Geometric Center, Front Longitudinal Margin, Rear Longitudinal Margin, and Longitudinal Stability Margin of the 9 Polygons Occurring in FTL Locomotion. (L is the length of the vehicle.)

Polygon	Geometric Center	Front Longit. Margin	Rear Longit. Margin	Longitudinal Stability Margin
(a)	(0,0)	0.4L	0.4L	0.4L
(b)	(-0.1L,0)	0.3L	0.3L	0.3L
(c)	(0.1L,0)	0.5L	0.5L	0.5L
(d)	(0.3L,0)	0.3L	0.3L	0.3L
(e)	(0.2L,0)	0.4L	0.4L	0.4L
(f)	(0.1L,0)	0.3L	0.3L	0.3L
(g)	(0.3L,0)	0.5L	0.5L	0.5L
(h)	(0.5L,0)	0.3L	0.3L	0.3L
(i)	(0.4L,0)	0.4L	0.4L	0.4L

$$X_{gc} = (0.3L+0.5L-0.5L-0.3L)/4 = 0$$

$$Y_{gc} = (-K+K-K+K)/4 = 0 \qquad (5.35)$$

The front longitudinal margin is the longitudinal distance from GC to the the front boder, $\overline{12}$, of the polygon. Every point lying on segment 12 satisfied the following equation:

$$\frac{0.5L - x}{x - 0.3L} = \frac{K - y}{y + K} \qquad (5.36)$$

The intersecting point of segment 12 and the horizontal line wich passes through the GC obviously has a Y coordinate of 0, hence its X coordinate can be easily calculated by substituting

$$y = 0 \qquad (5.37)$$

into Eq. (5.36), and the result is

$$x = 0.4L \qquad (5.38)$$

Consequently, the front longitudinal margin is 0.4L. The rear longitudinal margin can be found in exactly the same way and has a value of 0.4L. Since the front and rear margins are equal, the longitudinal stability margin is also 0.4L. Similar computation can be performed for every polygon to obtain the numbers shown in Table 5.1.

Note that the GC of each polygon is at different location, therefore the body has to move 8 times if it is required to move to every new GC. Moreover, by looking into the coordinates of each successive GCs, the direction of the motion of the body has the following sequence:

back, forth, forth, back, back, forth, forth, and back. Also note that the minimal longitudinal stability margin of the entire cycle is 0.3L.

It is interesting, at this point, to compare the longitudinal stability margins exhibited by the following gaits: the optimally stable wave gait, the regular symmetric gait 1-3-5-2-4-6 with constant body velocity, FTL gait using only three polygons, and FTL gait using 9 polygons. The comparisons were presented in Table 5.2.

In 1974, McGhee and Sun [61] determined the longitudinal margin for all regular symmetrical wave gaits for any even number of legs. According to their calculations, this quantity of a hexapod vehicle is 0.28L for a duty factor of 5/6, and is 0.35L for a duty factor of 1.

Sun has also computed the longitudinal stability margin for all regular symmetric gaits of a hexapod machine [39]. His result indicates that for the gait sequence 1-3-5-2-4-6 and a constant body velocity, the value is 0.1L for a duty factor of 5/6, and is 0.15L for a duty factor of 1.

For FTL gaits using 9 polygons, the resultant margin is 0.3L as illustrated in Table 5.1. If only three polygons are considered, the stability margin is 0.2L. In this case, when polygon (b) appears, the body does not move and the center of body remains at location (0,0). Recall from Table 5.1 that if the body moves into location (-0.1L, 0), the margin would be 0.3L in both directions. Thus with the body stays at (0,0), the front margin becomes 0.2L while the rear margin is 0.4L, which gives a stability margin of 0.2L.

To summarize, different motion algorithms based on the concept of OSP can be devised, depending the number of polygons that are taken into account, whether they are 5-vertex or 6-vertex polygons, and if backward

TABLE 5.2

Comparisons of Different Gaits in Terms of the Longitudinal Stability Margin.

(β is the duty factor and L is the length of the vehicle.)

Gait	Longitudinal Stability Margin (β = 5/6)	(β = 1)
Optimally Stable Wave Gaits	0.28L	0.35L
Regular, Symmetrical Gait (1-3-5-2-4-6) with Constant Body Velocity	0.1L	0.15L
FTL Gait Using 3 Polygons	0.2L	
FTL Gait Using 9 Polygons	0.3L	

motions are allowed. It should be emphasized again that the present software adopts a scheme which only employs three specific 6-vertex polygons in navigating the vehicle. This particular scheme indeed proves the feasibility of the FTL operation and was successfully implemented. However, it is by no means the only solution since many other motion algorithms are possible as explained above. As such, the introduction of the USP concept stimulates many interesting research issues in vehicle navigation which remain to be answered.

5.3.3 Heading Direction--What is the Turning Angle

When the center of the body is moving toward a new location as determined in the previous section, it can assume either a pure turning mode, a pure crabbing mode, or a combination of these two modes [5]. Generally speaking, the motion of body would involve X, Y velocities and a turning rate. While the linear velocities are easily computed by dividing the displacement vector with its traveling time, the determination of the turning rate requires some more thought. One approach is to store all the previous locations that the center of the body has traversed and to fit them by a polynomial so that the next movement can also be described by this polynomial. It takes large amount of memory space to store the trajectory and requires a great deal of computing time to update all points as the body moves around. Closer investigation indicates that it is not necessary to record all the previous positions since not all of them have the same effect on the upcoming movement. Fig. 5.22.a shows a motion trajectory which is varying constantly and the local behavior is quite different from the global pattern. It is obvious that local points have more influence in deciding the next action and are thus more important than other points. In Fig. 5.22.b, the local behavior agrees with the global one hence it is sufficient to use only recent points to prepare for next movement. Thus it was decided in this research to utilize three local points to specify the turning angle. They are: the next OSP, the present OSP, and OSP of the last cycle. The turning angle is computed as the intersecting angle between vectors $\overline{V1}$ and $\overline{V2}$ as shown in Fig. 5.23. In practice, a turning angle is not allowed to exceed a maximum value to avoid leg interference. If the computed value is larger than the acceptable one, it will only turn as much as it is permitted and try to make up the difference during the next coming cycle.

5.4 Checking for Kinematic Margin

During motion, if any leg goes out of its kinematic limits, the power of all joint motors must be turned off to avoid damage to the Hexapod, and the experiment is terminated abruptly. This constituted much of the author's frustration in performing experiments because (1) the phenomena that are of interest and are expected to be the main observations may be in the latter part of a locomotion, and (2) if Hexapod is unable to recover from an undesired posture through normalization or a manually operated battery box, it requires several persons to lift and move it by hand. With this concern, it is highly desirable to be able to check beforehand the joint angle that every joint may possess throughout the next cycle and report all cases that violate the specified limits. The mechanical limits of the joints are as follows:

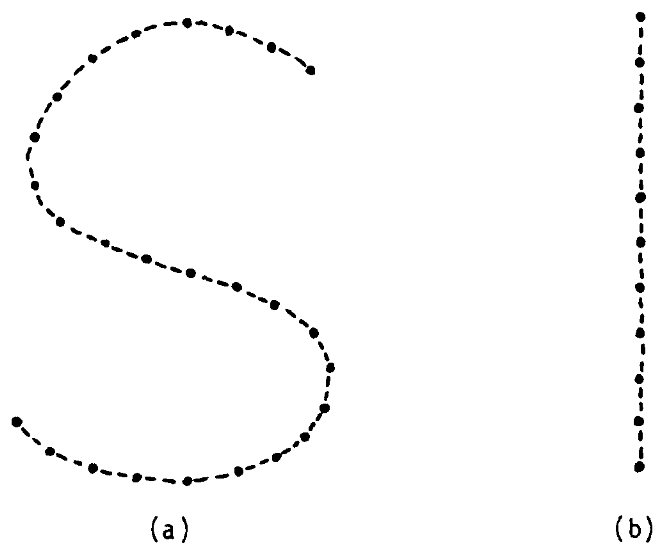


Figure 5.22 Possible Trajectories of the Center of the Body.
(a) Global Pattern Differs from Local Pattern;
(b) Global Pattern Agrees with Local Pattern.

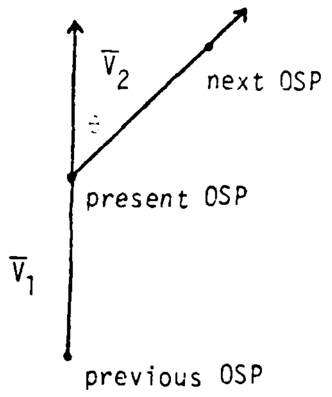


Figure 5.23 Determination of the Turning Angle.

$$|\theta_1| \leq 75°$$

$$|\theta_2| \leq 75°$$

$$|\phi| \leq 75°$$

where θ_1 is the angle of a elevation joint, θ_2 is the angle of a knee joint, and ϕ is the angle associated with an azimuth joint.

There is a one-to-one correspondence between the three joint angles of each leg and the linear coordinates (x,y,z) of the foot tip. Their relationship are expressed as a set of kinematic equations and inverse kinematic equations [56]. Once the next foothold is known, the corresponding joint angles are computed to see if any of them is out of bounds. Since footholds are represented in the body coordinate system, checking for kinematic margin during a cycle has to be carried out whenever the body moves into a new position. For the sequence in Fig. 5.3, checking would be performed at three occasions in a cycle as follows:

(1) The first checking is due upon the acquisition of a new foothold and before one of the front feet takes off. This allows the Hexapod to know whether a newly specified foothold is reachable within one step. If it it is too far away, the Hexapod would prompt the operator to assign another one. Note that this checking is needed not because the body has moved, but because a new foothold is given. Hence, only leg 1 or leg 2 needs to be checked, depending on which leg the foothold is for.

(2) The second checking is associated with the body motion after leg 3 is transferred to a new location. Here, all six legs are involved in checking since coordinates of their foot tips all change as the body

moves. By the optimal algorithm described earlier, the new location of the center of body can be computed, and each foot tip position after the body moves can be calculated in advance to foresee any abnormality.

(3) The third checking is similar to the second but is linked to the body motion after the action of leg 4.

In programming, all three checkings are "simulated" before real movement is initiated. If any violation is reported, the message "OUT OF REACH, TRY AGAIN !" will be displayed on the screen of a terminal to ask the operator to select another foothold. It should be emphasized again that this is possible because the trajectory of the center of body throughout a cycle can be predicted beforehand.

5.5 Update Coordinates of Footholds as Body Moves

Every foothold specified by the operator is used by all three legs on the same side in three consecutive cycles. This introduces the necessity to store and update coordinates of footholds for at least three cycles. Two different ways of dealing with this problem are delineated below.

(1) The open-loop approach. In this case, each foothold is treated in exactly the same way as a supporting leg is handled in terms of the computation of new coordinates as seen from the body coordinate system. Since the velocities and turning rate are known, the calculation may be done based on the "relative motion" concept. Mathematical analysis of this part was provided by Wahawisan in [56]. This method is considered to be an open-loop way because all computations utilize commanded body

velocities and turning rate as the primary parameters without verifying if those commanded values have really been reached. If the body follows them closely, a good result is obtained; otherwise, errors in foothold coordinates tend to accumulate from cycle to cycle.

(2) The close-loop approach. In contrast to method (1), actual body motion instead of commanded body motion is used here to update footholds. Actual body motion on even terrain is characterized by (dx, dy, θ), where dx and dy are the displacement components in X and Y directions and θ is the turning angle. These quantities are measured at the new body position with respect to the old body coordinate system, as illustrated in Fig. 5.24. Values of dx, dy and are computed by using the actual foot tip positions of two supporting legs at instants t1 and t2, where t1 corresponds to the time when body assumed the old position and t2 is associated with the new location. The following derivation shows how they are calculated.

Suppose that during the time interval [t1,t2] the center of body moved (dx,dy) and turned an angle of θ. The geometrical relationship between the new and old coordinate systems may be expressed as a HTM ${}^{O}T_N$ where

$$ {}^{O}T_N = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & dx \\ \sin\theta & \cos\theta & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.39} $$

The coordinates of a foothold in both CS's are related by ${}^{O}T_N$ in the following way:

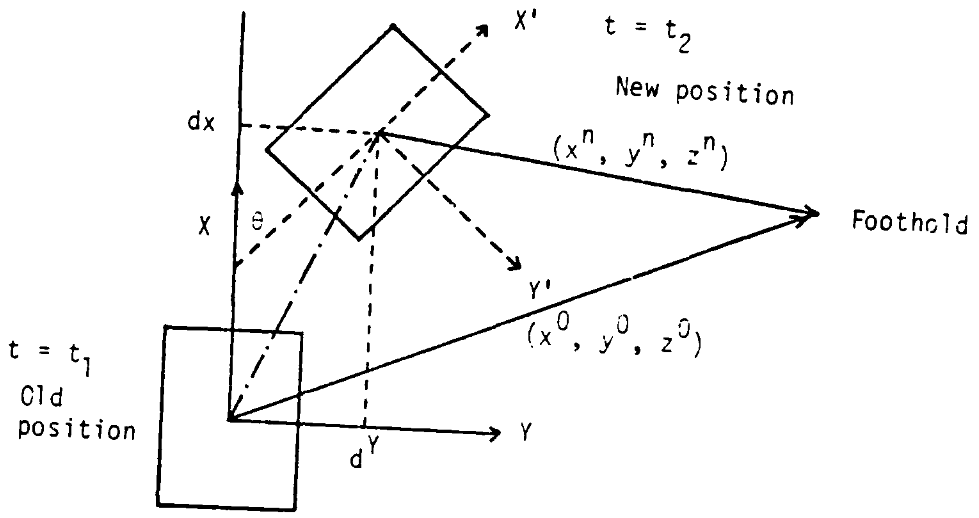


Figure 5.24 Foothold Locations as Seen from the Old and the New Body Positions.

$$\begin{bmatrix} X_i^O \\ Y_i^O \\ Z_i^O \\ 1 \end{bmatrix} = {}^{O}T_N \begin{bmatrix} X_i^N \\ Y_i^N \\ Z_i^N \\ 1 \end{bmatrix} \tag{5.40}$$

where the superscript 'n' and 'o' denote new and old CS's, and the subscript i indicates it is the ith leg.

Performing multiplication on the right hand side and equating the corresponding row, two equations are obtained:

$$\begin{aligned} X_i^o &= \cos\theta \cdot X_i^N - \sin\theta \cdot Y_i^N + dx \\ Y_i^o &= \sin\theta \cdot X_i^N + \cos\theta \cdot Y_i^N + dy \end{aligned} \tag{5.41}$$

Therefore, during that period, if both leg 1 and leg 2 are in the supporting phase, then four equations are available:

$$\begin{aligned} X_1^o &= \cos\theta \cdot X_1^N - \sin\theta \cdot Y_1^N + dx, \\ Y_1^o &= \sin\theta \cdot X_1^N + \cos\theta \cdot Y_1^N + dy \\ X_2^o &= \cos\theta \cdot X_2^N - \sin\theta \cdot Y_2^N + dx \\ X_2^o &= \sin \cdot X_2^N + \sin\theta \cdot Y_2^N + dy \end{aligned} \tag{5.42}$$

Combining the similar terms in Eq. (5.42) results in

$$(X_1^O - X_2^O) = \cos\theta(X_1^N - X_2^N) - \sin\theta(Y_1^N - Y_2^N) \tag{5.43}$$

$$(Y_1^O - Y_2^O) = \sin\theta(X_1^N - X_2^N) + \cos\theta(Y_1^N - Y_2^N) \tag{5.44}$$

Multiplying Eq. (5.43) with $\cos\theta$ and Eq. (5.44) with $\sin\theta$ yields

$$\cos\theta\ (X_1^O - X_2^O) = \cos^2\theta(X_1^N - X_2^N) - \cos\theta \cdot \sin\theta(Y_1^N - Y_2^N)$$

$$\sin\theta\ (Y_1^O - Y_2^O) = \sin^2\theta(X_1^N - X_2^N) + \sin\theta \cdot \cos\theta(Y_1^N - Y_2^N) \tag{5.45}$$

Adding them together leads to

$$(X_1^O - X_2^O) \cdot \cos\theta + (Y_1^O - Y_2^O) \cdot \sin\theta = (X_1^N - X_2^N) \tag{5.46}$$

Let

$$A = (X_1^N - X_2^N)$$

$$B = (X_1^O - X_2^O)$$

$$C = (Y_1^O - Y_2^O) \tag{5.47}$$

Substituting them into Eq. (5.46) gives

$$B \cdot \cos\theta + C \cdot \sin\theta = A \tag{5.48}$$

By taking out a common factor, Eq. (5.48) can be rewritten as

$$\sqrt{B^2 + C^2} \cdot \left(\frac{B}{\sqrt{B^2 + C^2}} \cos\theta + \frac{C}{\sqrt{B^2 + C^2}} \sin\theta \right) = A \tag{5.49}$$

Let

$$\sin\phi = \frac{B}{\sqrt{B^2 + C^2}}$$

$$\cos\phi = \frac{B}{\sqrt{B^2 + C^2}} \qquad (5.50)$$

Eq. (5.49) becomes

$$\sin\phi \cos\theta + \cos\phi \sin\theta = \frac{A}{\sqrt{B^2 + C^2}} = K \qquad (5.51)$$

Hence

$$\sin(\phi + \phi) = K \qquad (5.52)$$

$$\phi + \phi = \sin^{-1} (K) \qquad (5.53)$$

Since $\tan\phi = B/C$

thus,

$$\phi = \tan^{-1} (B/C) \qquad (5.54)$$

Therefore,

$$\theta = \sin^{-1}(K) - \tan^{-1}(B/C) \qquad (5.55)$$

The values of dx and dy then can be computed by using Eq. (5.42).

5.6 Contact Sensing

When a leg is placed down at the end of a transfer phase, it may fail to reach the terrain surface because (1) the foothold position provided by the vision system has a significant error in the Z direction, or (2) the servo loop fails to produce the commanded movement. This may introduce undesirable consequence for the Hexapod. For instance, considering the FTL sequence, leg 3 is supposed to take off as soon as leg 1 touches down. If leg 1 does not make contact with the ground while leg 3 lifts up as scheduled, then on the left side of the Hexapod only leg 5 is actually in the supporting phase and shares the weight of the vehicle. Leg 5 would be overloaded and the machine becomes unbalanced. Therefore, it is very desirable to be able to guarantee that a leg truly reaches ground when it is coming down. With the force sensor installed inside each leg, this can be accomplished by feeding back the reacting force and checking if its value exceeds a threshold, say, 15 pounds. In cases that a transfer phase expires and the reaction force is still smaller than the threshold, the leg will be commanded to extend its motion in the Z direction; meanwhile the leg scheduled to lift up next will be delayed until the force sensor indicates that contact has been made for the preceding leg. This contact testing has been included in the author's software WALK37 (see Appendix).

Another possible application of force sensors, which remains to be implemented, is to determine if the terrain surface is stiff enough to support the load of a leg. For example, if a foothold is assigned on the surface of quicksand, force fed back from the leg should indicate, to some degree, the texture of that foothold. If no substantial force is sensed after a period of searching, it would be concluded that the point is not solid enough for a leg to step on and another one should be selected.

In addition, a force sensor can be used to realize active compliance [62]. This subject is covered in Section 6.3.

5.7 Overview of FTL Locomotion

The motion algorithm described in this chapter is summarized into the flowchart shown in Fig. 5.25. The operator initializes the first half motion cycle by assigning a foothold for leg 1. This foothold is perceived by the vision system and its coordinates are computed in the 11/70 computer using triangulation. With this information, the next, 6-vertex supporting polygon can be identified, its optimally stable point can be computed, and the motion of the body can be pre-planned. A checking procedure is then invoked to simulate the planned motion and to see if any leg would go out of its kinematic limits. The foothold would be rejected for any violation of kinematic margin and the operator would be asked to assign another point. After a foothold is accepted, leg motion begins. Leg 1 moves first, followed by leg 3, then the body moves, and then leg 5 is moved. These events constitute the activities of the first half cycle. The scenerio of the second half cycle is exactly the same except that it involves legs 2, 4, and 6 instead of 1, 3, and 5. A new cycle then starts and the whole operation is repeated.

5.8 Summary

Algorithms which have been developed to facilitate FTL motion were presented in this chapter. Activities that occur at the locomotion-cycle level were discussed in detail. A particular gait sequence, 1-3-5-2-4-6, was selected to implement FTL operation because it satisfies the requirement of FTL motion and possesses the property of physical symmetry. The behavior of the body was investigated with respect to timing, destination, and heading direction. The body was scheduled to move after the placement of the middle leg and before the placement of

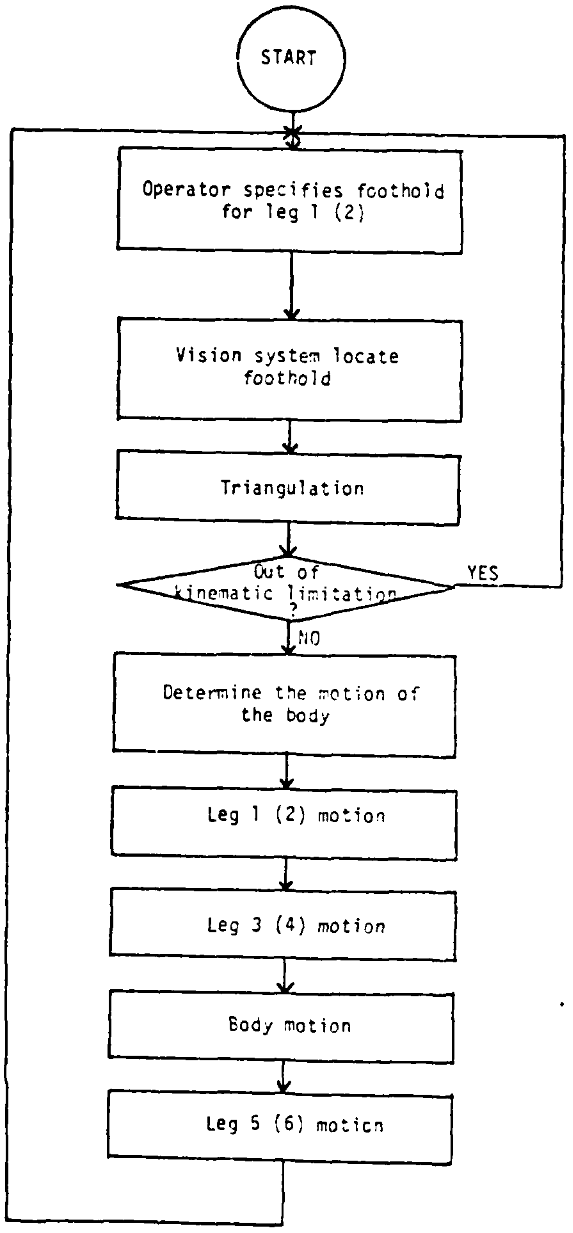

START
Operator specifies foothold for leg 1 (2)
Vision system locate foothold
Triangulation
Out of kinematic limitation ?
YES
NO
Determine the motion of the body
Leg 1 (2) motion
Leg 3 (4) motion
Body motion
Leg 5 (6) motion

Figure 5.25 Flowchart of FTL Walking Algorithms.

a rear leg. All six legs are in the supporting phase during body motion to enhance stability. The destination of the body was chosen as the the OSP of the next, 6-vertex supporting polygon, and only three such polygons were considered in the current software. The heading direction was determined bu using Three consecutive OSPs. It should be point out that the geometric center of all six legs is used in place of the OSP for consideration of real-time programming.

A heuristic algorithm was presented to find the OSP of an arbitrary polygon. For simple polygons such as regular polygons or BIP, the algorithm leads to the same result as produced by the geometrical approach, which locates the OSP at the intersection of all angular bisectors. In general, the shrinking operation has to be applied iteratively to identify OSP. This process is rather complicated and time-consuming, thus is not included in the present version of software (the geometric center is used instead of the OSP). However, recent study by Miller [63] indicates that array processors can efficiently perform "image shrinking", which has a similar geometric nature to the shrinking of a polygon. Hence, the computation of OSPs may become feasible for real-time control if array processors are employed.

Through the realization of FTL motion, it is also recognized that what is really needed is a body servo planning routine which navigates from one OSP to the next for every supporting pattern while maximizing stability margin over the entire trajectory and rotating the body to maximize kinematic margin. Compared with this extremely complicated and general problem, this chapter deals a much simplified case in a particular way. More research effort is needed in this area.

Chapter 6

ROUGH TERRAIN LOCOMOTION

6.1 Introduction

Legged vehicles are designed to walk over rough terrain; the Hexapod is no exception. When moving on an even surface, it may be assumed that the surface condition does not change, that every point on the ground is a good foothold, and that footholds for all legs lie on the same plane. For such a simple case, there is little need for the Hexapod to sense and analyze the terrain; all it has to do is to recognize and to locate those specified footholds. In fact, even the Z coordinate of a foothold is redundant since the Hexapod knows that it should lie on the horizontal surface. However, traversing rough terrain is a different matter. Rough terrain may be defined as a three-dimensional space in which the heights (or Z coordinates) of surface points are not constant. Therefore, slopes, ditches, holes, obstacles, etc., may exist. Obviously, the Hexapod needs more external sensors to overcome such a complicated environment. This is why it was restrained to walk only on even terrain until force sensors and a vertical gyro were installed. In recent experiments, Pugh demonstrated that, with the assistance of these devices, the Hexapod was able to stably traverse a pile of wooden logs while maintaining level body motion [46]. Two things should be pointed out in regard to his work. First, it was accomplished before the vision system was constructed, and proved that rough terrain locomotion was possible even without the vision function. Second, since vision was not involved, no prior information about the terrain lying ahead was available, and everything was treated in a feedback-and-correct manner, i.e.; adjustments were performed only after something had happened.

With a vision system, the Hexapod knows where the next foothold is before it makes the subsequent movement. This allows it to prepare and to make proper adjustment to accommodate the variation of terrain surface. Ideally speaking, if the vision system is perfect, then the Hexapod should be able to regulate itself adequately even without force sensors or gyros. Unfortunately, like every real system, the vision system indeed has a certain amount of error. Hence, to ensure safety and a smooth ride, algorithms developed by Pugh are incorporated with the vision system in resolving rough terrain locomotion.

Since it is intended to encompass rough terrain locomotion by extending those algorithms described in Chapter 5 which were developed under the assumption that the Hexapod body is parallel to a horizontal plane, one of the central problems in dealing with uneven terrain is to control its attitude so that the body is maintained level. A detailed treatment of this issue has been given in [64] and [65]. Thus only a qualitative summary is presented in Section 6.2. Active compliance is especially important for rough terrain motion because it effectively provide an adjustable suspension system and can reduce much of the transient impact when something unexpected occurs. The theoretical background and mathematical derivation of this subject was delineated in

[65]. Section 6.3 of this chapter attempts to give a qualitative interpretation of the effect of active compliance. Section 6.4 discusses the subject of altitude estimation which is needed to update the Z coordinate of footholds for rough terrain locomotion.

6.2 Attitude Control

The OSU Hexapod is a very complex machine. While the body alone has six degrees of freedom in motion (three in translation and three in rotation), each of the six legs can move independently in three directions, thus adding another three degrees of freedom to the overall motion of the entire system. Therefore, the total number of generalized coordinates for a rigid body model is 24, impling a state vector of dimension 48 when position and velocity are considered [64]. Previous studies have shown that, for such a complicated system, an analytical approach for regulation of the movements is not feasible, even if small motion linearization is assumed [66]. Hence, the control of body attitude discussed here is based on a much simplified model which treats individual supporting legs separately. Precisely, attitude control is achieved through adjustment of the effective length of all supporting legs, as illustrated in Fig. 6.1 and Fig. 6.2.

Fig. 6.1 shows the side view of the situation where the Hexapod has a pitch angle of P radians. If the length of leg F (front leg) is decreased by Δz_F and that of leg R (rear leg) is increased by Δz_R, as seen in the figure, the body would become level. By simple trigonometry,

$$\Delta z_F = -X_F \cdot \text{Tan } P \qquad (6.1)$$

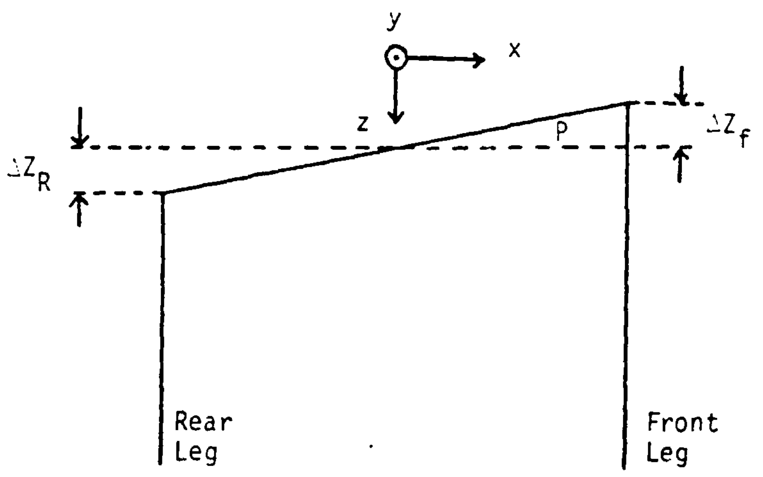


Figure 6.1 Side View of Hexapod with a Pitch Angle.

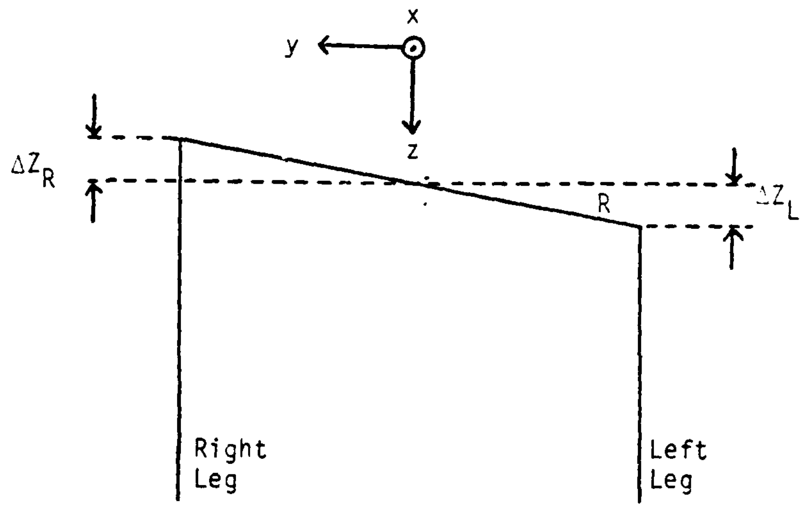


Figure 6.2 Front View of Hexapod with a Roll Angle.

where X_F is the x coordinate of the foot tip of leg F as expressed in the body coordinate system. A positive value of Δz_F indicates that the length of this leg should be increased rather than decreased.

Assume that attitude control is efficient so that the pitch angle is always kept within a small value, then tan P approaches P and

$$\Delta zF = -X_F \cdot P \tag{6.2}$$

For the rear leg, a similar analysis leads to

$$\Delta z_R = -X_R \cdot P \tag{6.3}$$

In general, the amount of adjustment of leg length Δz_i of leg i is

$$\Delta z_i = -X_i \cdot P \tag{6.4}$$

Note that the sign of Δz_i is decided by that of X_i and the pitch angle P, which is positive in the counterclockwise direction about the Y axis. For instance, in Fig. 6.1, P is positive, X_F is positive but X_R is negative, therefore dz_F is negative and dz_R is positive.

The correction of leg length needed to overcome a roll angle R can be derived in a similar way, and the result is

$$\Delta z_i = Y_i \cdot R \tag{6.5}$$

where Y_i is the y coordinate of the foot tip of leg i expressed in the body coordinate system.

For small pitch and roll angles, the foot adjustments needed for an error around a given axis is independent of the error around the other. Thus, Eqs. (6.4) and (6.5) can be combined together to deal with the cases where both pitch and roll angles are present. The result is simply

$$\Delta z_i = -X_i \cdot P + Y_i \cdot R \tag{6.6}$$

In additional to adjusting the Z position, the rate in the Z · direction, Z, also has to be modified because a resolved rate control mechanism [44] is adopted for the servo control loop. This quantity is obtained by differentiating Δz_i of Eq. (6.6) with respect to time, which gives

$$(\dot{\Delta z_i}) = -X_i \cdot \dot{P} + Y_i \cdot \dot{R} \tag{6.7}$$

If one allows the pitch and roll angles to decay exponentially rather than trying to force them to zero immediately, which is virtually impossible for a physical system, the following relations hold:

$$\dot{P} = -K \cdot P$$

$$\dot{R} = -K \cdot R \quad , \quad K > 0 \tag{6.8}$$

Substituting Eq. (6.8) into Eq. (6.7) yields

$$(\dot{\Delta z_i}) = -K(-X_i \cdot P + Y \cdot R) \tag{6.9}$$

and Δz_i can be computed in real time by integrating $(\dot{\Delta z_i})$ over a time interval, giving

$$\Delta z_i = \int_{ts}^{t} (\dot{\Delta z_i}) \cdot \Delta t \tag{6.10}$$

where ts is the instant at which leg i enters the support phase.

The resultant Δz_i and $(\Delta \dot{z}_i)$ are then respectively superposed to the corresponding component generated by the foot trajectory algorithm [46] to form the desired Z position and the desired Z rate, as shown in the following equations.

$$Z_i^D = \Delta z_i + Z_i^T$$

$$\dot{Z}_i^D = (\Delta \dot{z}_i) + \dot{Z}_i^T \tag{6.11}$$

where Z_i^T and $\dot{Z}_i^T$ are the outputs from the foot trajectory loop, and Z_i^D and $\dot{Z}_i^D$ are the desired values.

6.3 Active Compliance

As mentioned in Section 5.6, a force sensor can be used as a contact sensing device to assure that a leg actually reaches the ground when it is placed down. Another important application of a force sensor is to implement active compliance. This is made possible by feeding back actual position, actual rate, and actual ground reacting force into a rate control loop, as described in the following equation [62]:

$$\dot{Z}_D = \dot{Z}_A + K_p (Z_D - Z_A) + K_f(F_A - F_D) \tag{6.12}$$

Those variables with subscript "D" denote desired values and those with subscript "A" represent actual values.

By assuming an ideal rate servo loop (the case of a nonideal loop was discussed in [46]), i.e., $\dot{Z}_A = \dot{Z}_D$, Eq. (6.12) becomes

$$(F_A - F_D) = -K_p/K_f \cdot (Z_D - Z_A) = K \cdot (Z_D - Z_A), \quad K > 0 \qquad (6.13)$$

Note that Eq. (6.13) may be interpreted as the equation of a spring, with K as the spring constant, Z_D the length of the relaxed spring, F_A the external compressing force, and F_D the external stretching force.

Therefore, with active compliance, a leg would behave as if a mechanical spring with an adjustable center were built inside it. This interpretation leads to the model shown in Fig. 6.3, which provides a good insight into the nature of active compliance and makes the makes the explanation of its effect easier. Here, a leg is modeled as consisting of a spring and a rigid segment, which means that forces acting on the foot tip will be transferred to the spring. When a leg contacts the ground, a compressing force F_A is exerted at the foot tip. The stretching force F_D is imaginary and is generated by programming. If the magnitude of F_A is greater than that of F_D, the spring is compressed and the leg becomes shorter. On the other hand, if F_D is greater than F_A, tne spring is stretched and the leg becomes longer.

The merits of active compliance can be fully appreciated considering the following two cases:

(1) Suppose that at the end of a transfer phase, a leg does not reach the ground due to errors or irregular terrain (say, a hole). However, during the following motion planning cycle, it is treated as a leg in the supporting phase and is required to share part of the weight of the Hexapod. Thus, a nonzero F_D is assigned by the program to this leg. Since it does not make contact with the ground, the value of F_A is zero. Consequently, F_D is greater than F_A and the leg will be extended

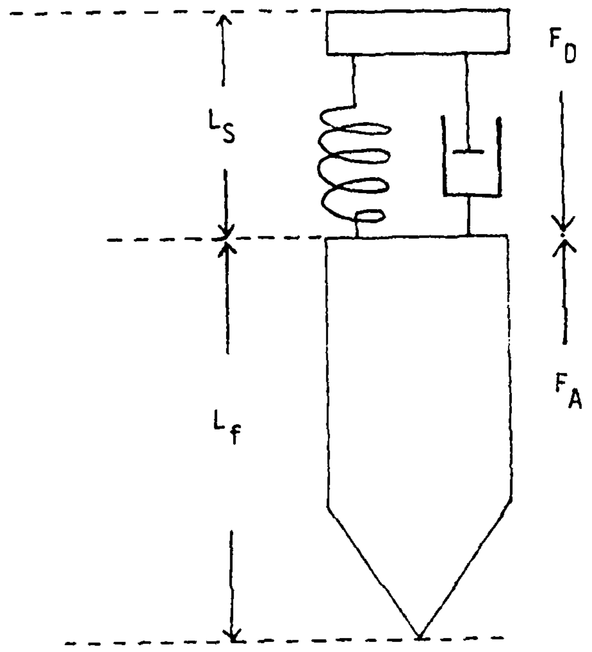


Figure 6.3 A Model for a Leg with Active Compliance [46].

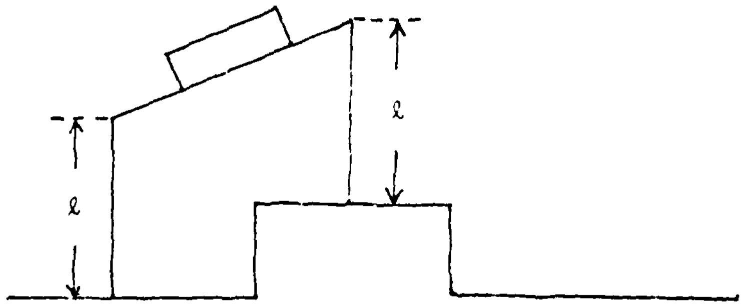


Figure 6.4 Excessive Body Tilting due to a Lack of Active Compliance.

further until it finally touches the ground, or the the term from position error and that from force error cancel out. If the hole is not too deep, this can ensure that a supporting leg always firmly contacts with the ground.

(2) Imagine that a leg is moving downward during a transfer phase and an obstacle is encountered before time expired. Since, according to the timing, the leg is considered as in the transfer phase, a zero valued F_D is assigned by the program. Meanwhile, a compressing force is exerted on the foot tip, causing F_A to be greater than F_D. This in turn will reduce the effective length of the leg. Notice that if it were not for the active compliance, the leg would have the same length as other legs, which would tilt the body excessively as shown in Fig. 6.4.

6.4 Altitude Estimation

When the Hexapod is walking on even terrain and the length of all legs is commanded to have the same value throughout the motion, it is reasonable to assume that the height of the body, which is measured from the center of the body to the terrain surface, will not change significantly. However, as the terrain surface fluctuates from point to point and the attitude control algorithm is imposed to keep the body level, the effective length of each leg is subject to change. It is conceivable that the altitude of the body would vary constantly as the Hexapod tries to accommodate itself to rough terrain.

Altitude estimation is needed due to the necessity of updating every foothold for at least three cycles. The analysis in Section 5.5 deals with the cases where the height of the body is invariant and thus the issue reduces to a two-dimensional problem, i.e., updating the X and Y coordinates of each foothold. Now that the altitude may change from time to time, it is also necessary to update the Z component as well. However, so long as the Hexapod is maintained level, the updating of the Z coordinate is independent of that of X and Y components; hence the result from section 5.5 is still applicable and the only additional work is to recompute the Z coordinate of each foothold. Since the Z coordinate is expressed in the body coordinate system with the center of the body as the origin, a change in body altitude implies a variation in the Z component of every foothold. Thus updating the Z coordinate is essentially the same as estimating the altitude of the body. For rough terrain locomotion, the Z coordinate of the foot tip of each leg is generally different from one another, and the altitude of the body can be defined as the average value of the Z components of all supporting legs. Hence, during the time period [t1,t2], the height of the body at t1, H[t1], is computed as

$$H[t_1] = \frac{\sum_{i=1}^{N} Z[i,t_1]}{N} \tag{6.14}$$

where N is the number of supporting legs at t_1. Similarly, $H[t_2]$ is calculated as

$$H[t_2] = \frac{\sum_{i=1}^{N} Z[i,t2]}{N} \tag{6.15}$$

and the Z component of a foothold at t_1 and t_2 are related by the following equation:

$$FTZ[i,t_2] = FTZ[i,t_1] + (H[t_2]-H[t_1]) \tag{6.15}$$

where $FTZ[i,t_2]$ is the Z coordinate of leg i at t1 and $FTZ[i,t_2]$ is that of the same leg at t_2. Therefore, knowing $FTZ[i,t_1]$ and the body altitude at t_1 and t_2, $FTZ[i,t_2]$ can be computed.

6.5 Summary

The following modifications were made to algorithms developed in Chapter 5 to deal with rough terrain locomotion: (1) foot lift was increased from 4 inches to 10 inches so that a leg could transfer across obstacles without collision, (2) body attitude control was imposed to keep the body level to a horizontal plane so that motion in XY plane and that in Z direction were decoupled, (3) active compliance was added to provide a suspension system so that transient impact was minimized, and (4) altitude estimation was performed to update the Z coordinate of a foothold.

The autnor would like to emphasize again that the subjects of attitude control and active compliance have been investigated previously in detail by Pugh, and the results were well documented in [46]. Since they were incorporated into the author's software to help resolving the problem of rough terrain motion, brief descriptions were included in this chapter for the completeness of this dissertation.

Chapter 7

EVALUATION OF THE VISION SYSTEM AND EXPERIMENTAL RESULTS

7.1 Introduction

The goal of FTL operation is to enable the Hexapod to walk over even/rough terrain under the guidance of a human operator. Specifically, the front feet should step onto the footholds assigned by the operator, the middle legs should follow the footprints of the front legs, and the rear legs must follow the steps of the middle feet. Therefore, the overall performance of FTL operation depends on three factors: (1) the accuracy of the vision system, (2) the accuracy of the servo control loop, and (3) the precision by which a foothold is updated from one locomotion cycle to another. While (1) and (2) determine how close a front foot can step onto a specified foothold, factors (2) and (3) dictate whether a rear or a middle leg can truly follow their predecessors. In Section 7.2, the vision system is evaluated in terms of its accuracy, cost and reliability. Experimental results of a typical FTL operation are given and discussed in Section 7.3.

7.2 Evaluation of the Vision System

7.2.1 Accuracy

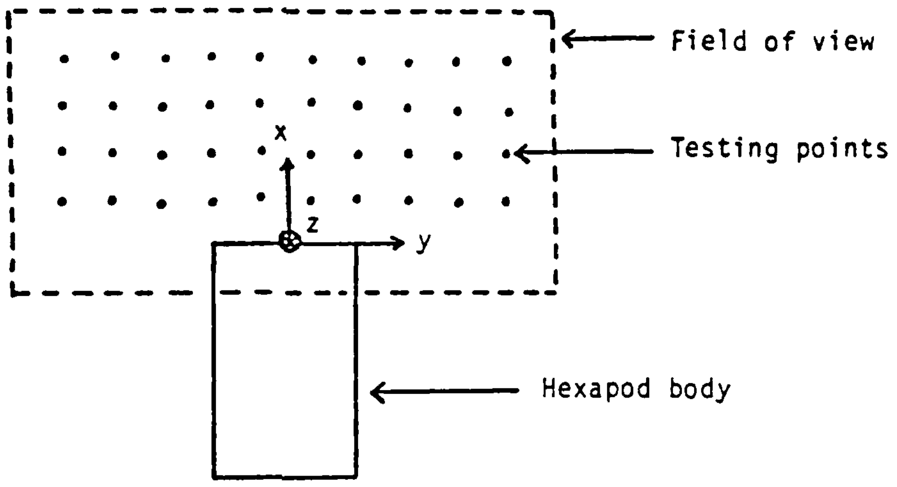
To find out how precise the vision system is capable of locating a foothold, the following test was performed. With the Hexapod standing at a fixed location, a number of points inside its field of view were selected and their coordinates were measured manually with respect to the body coordinate system. The laser beam was then pointed at the selected spots, one at a time, the vision system was activated to sense the image, and the coordinates were computed by triangulation. This arrangement is shown in Fig. 7.1. The results are tabulated in Table 7.1, where RX, RY, and RZ represent the coordinates measured by hand; IX1, IY1, IX2, IY2 are the image coordinates decoded by the interface circuit; and CX, CY, CZ are the coordinates computed using triangulation. For the 35 entries of data shown in the table, the average errors of the X, Y, and Z coordinates are 1.34 cm, 1.16 cm, and 1.2 cm, respectively.

The locating of a testing point may also be considered as the determination of a vector from the middle of the two cameras to the testing point. The length of this vector, ℓR, is

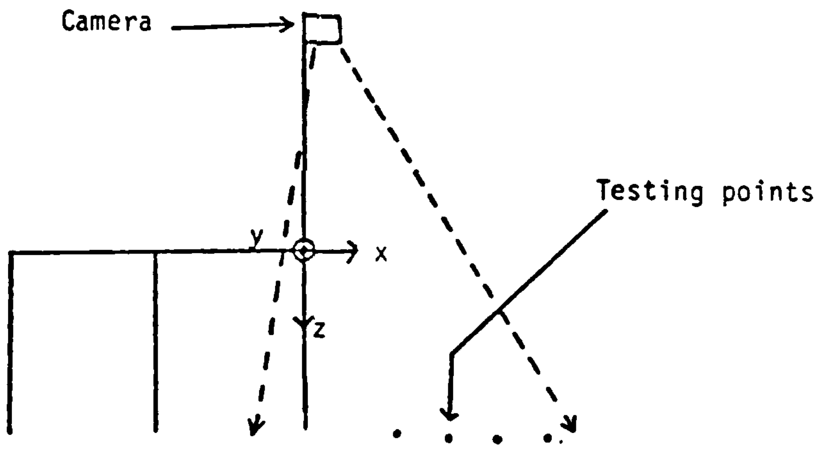
$$\ell R = \sqrt{(RX)^2 + (RY)^2 + (RZ+50)^2} \tag{7.1}$$

An offset of 50 cm is added to RZ because the cameras are 50 cm higher than the reference point with respect to which measurements are performed. The error of a measurement, $\Delta\ell$, is

$$\Delta\ell = \sqrt{(RX-CX)^2 + (RY-CY)^2 + (RZ-CZ)^2} \tag{7.2}$$



(a)



(b)

Figure 7.1 Arrangement for Collecting Triangulation Data. (a) Top View; (b) Side View.

TABLE 7.1
Data of Triangulation Experiments.

(RX, RY, and RZ are the actual coordinates of a testing point; IX1, IY1, IX2, and IY2 are the image coordinates; CX, CY, CZ are the computed values by triangulation.)

	RX (cm)	RY (cm)	RZ (cm)	IX1	IY1	IX2	IY2	CX (cm)	CY (cm)	CZ (cm)
1	24	90	77	198	151	207	123	25.65	87.7	77.43
2	24	67.5	77	177	150	190	126	25.15	64.7	76.1
3	24	44	77	154	150	170	130	24.5	43.27	78.09
4	24	21	77	128	147	147	134	24.17	19.93	77.92
5	24	0	77	106	144	125	138	24.09	-0.2	79.7
6	24	-24.5	77	78	142	97	142	23.09	-25.85	78.99
7	24	-47.5	77	56	139	72	145	22.54	-48.	78
8	24	-70.5	77	36	135	48	146	23.1	-72.4	79.56
9	24	-93	77	19	132	27	148	22.54	-95.6	79.57
10	46	90	77	197	123	201	96	49.5	89.03	78.67
11	46	67.5	77	176	120	184	99	48.15	64.67	76.67
12	46	44	77	154	118	165	101	47.73	42.8	76.53
13	46	21	77	130	116	143	103	47.48	20.13	76.57
14	46	0	77	107	114	121	107	45.79	-1.46	74.66
15	46	-24.5	77	82	111	95	111	46.5	-26.13	77.78
16	46	-47.5	77	61	110	72	114	45.4	-47.6	77.01
17	46	-70.5	77	42	108	50	116	45.25	-69.6	76.33
18	46	-93	77	25	106	29	119	45.52	-93.8	77.7

TABLE 7.1 - continued

	RX (cm)	RY (cm)	RZ (cm)	IX1	IY1	IX2	IY2	CX (cm)	CY (cm)	CZ (cm)
19	45.5	85	74	195	121	200	96	48.7	84.6	74.87
20	45.5	62.5	74	174	119	183	98	47.18	60.9	72.73
21	45.5	40	74	150	117	163	101	47.10	38.62	74.38
22	45.5	17	74	125	115	139	104	46.18	15.37	73.44
23	45.5	0	74	107	113	121	106	46.27	-1.46	73.88
24	45.5	-29	74	77	110	91	112	44.81	-29.49	73.5
25	45.5	-51.5	74	55	110	66	115	43.4	-51.94	72.78
26	45.5	-74.5	74	37	108	45	119	43.01	-73.68	73.6
27	45.5	-97	74	20	107	24	122	42.5	-97.53	74.28
28	23	85	74	197	150	207	122	25.19	83.45	71.9
29	23	62.5	74	175	151	189	126	24.07	61.4	72.73
30	23	40	74	150	149	167	131	24.22	39.45	76.75
31	23	17	74	124	147	144	135	23.16	16.31	74.97
32	23	0	74	104	145	125	139	21.98	-1.09	73.13
33	23	-29	74	73	142	93	144	21.68	-29.18	75.59
34	23	-51.5	74	51	139	67	147	21.05	-51.47	74.54
35	23	-74.5	74	31	136	44	148	20.2	-72.89	71.83

The average error, AE, of the 35 tests is

$$AE = \frac{\sum_{i=1}^{35} |\Delta \ell_i|}{35} = 2.3688 \text{ (cm)} \tag{7.3}$$

while the average vector length, AVL, is

$$AVL = \frac{\sum_{i=1}^{35} \ell R_i}{35} = 140 \text{ (cm)} \tag{7.4}$$

and the percentage error, E, is

$$E = \frac{AE}{AVL} \times 100\% = 1.69\% \tag{7.5}$$

7.2.2 Cost

The components comprising the vision ystem and their individual prices are listed in Table 7.2.

TABLE 7.2
Components and Costs of the Entire Vision System

Item	Quantity	Cost
Camera and CCU	2	$7000.00
CRT monitor	1	670.00
45 ft. cable	2	450.00
Fisheye lens	2	330.00
Optical filter	2	302.00
Laser head	1	400.00
Laser power supply	1	250.00
Ball socket head	2	40.00
Interface board	1	300.00
IC chips	86	100.00
Power supply unit	1	100.00
TOTAL		$9942.00

The total cost of the entire vision system is nearly $10,000., with the cameras and their CCU's contributing to two thirds of the total amount. The optical radar system constructed by ERIM is estimated to be about $200,000., which is 20 times as expensive as this vision system [47]. Moreover, except for the special purpose interface circuit, most of the components can be used for other applications. For example, the laser unit may be utilized for experiments which require a laser beam as a source, and the CRT monitor can be connected to any standard video signal generator as a display instrument. By adding more equipment, for instance, a simple image processor which can store one frame of image data, the potential of the TV cameras can be further exploited as a useful tool for research work in the areas of image processing and pattern recognition. Hence, the vision system not only is a crucial element for the successful implementation of FTL operation, but also provides possible opportunities for other research. As such, its cost is considered worthwhile and well-justified.

7.2.3 Reliability

At the time of this writing, the binocular vision system has been installed and operated for the past 6 months. Not a single case of malfunction, either in hardware or in software, has been observed so far. It is thus concluded that the whole system is highly reliable.

7.2.4 Operational Speed

After a foothold is selected by the operator, it takes three steps to obtain the spatial coordinates. First, the hardware circuit needs time t_1 to decode the image coordinates. Second, it takes time t_2 to transfer the 5 bytes of data to the computer. Third, the triangulation software needs time t_3 to compute the coordinates. Since the cameras are operated at a frequency of 30 frames per second and a complete image detecting cycle takes up to 4 frames (since there are 4 states for each image decoder), t_1 is approximately 120 msec. Using the maximum baud rate, 9600, t_2 is 5 msec. The triangulation subroutine takes 0.6 msec. to perform the computation. Thus, the total time needed is 125.6 msec.

Obviously, the bottle neck is in the image detecting phase. Nevertheless, the vision system is only activated twice during a locomotion cycle to acquire image data, and the current operational speed is more than sufficient. If it is to be used to collect terrain data and to build a map point by point, higher speed is desirable. Improvement is possible by employing a high speed camera, such as the one used in the Ohio State University Gait Laboratory [67], and by parallel data transmission between the interface circuit and the 11/70 computer.

7.2.5 Limitations of the Vision System

There are a few limitations on the vision system:

(1) It will detect only one image for the whole frame. Since it is decided that only one foothold will be assigned at a time, as far as FTL operation is concerned, this is not considered a disadvantage. Multiple image detection would become possible if more hardware were to be added to the existing interface board.

(2) The laser beam must be projected at points which are sufficiently close to the cameras so that the intensity of the laser image is strong enough to activate the photo sensors. In other words, there is a limited range within which a laser spot will be sensed. Furthermore, the light source should be inside the common field of view of the cameras. It turns out that the latter requirement overrides the former one, i.e., the laser image must be projected within the 2 m by 2 m field of view.

(3) The laser beam must be pointed on an object which will reflect enough light to be picked up by the cameras. For example, if the laser beam is pointed to a black object, no image can be detected by the system and FTL operation will not be completed.

7.3 Experimental Results

To monitor if each leg really steps onto its foothold, data of the actual foot positions and the foothold locations are recorded and stored when the Hexapod is in motion, and then examined and analyzed off-line. Fig. 7.2 through Fig. 7.7 represent typical results obtainable from a FTL operation for a complete cycle. Three figures are related to each leg, representing the X, Y, and Z coordinates, respectively, of the foot tip position and the foothold location.

Fig. 7.2 shows the X component of the actual foot tip position of leg 1, noted as XPA[1], and the corresponding component of the assigned foothold for it, FTHD[1,x], as functions of time. Both are expressed in the body coordinate system. Initially, XPA[1] has a value of 14 inches, and the specified foothold is 32 inches away from the center of the body. During period [0,a], leg 1 is moving forward to its destination. As can be seen from the figure, the solid line meets the dashed line at point a, indicating that leg 1 indeed reaches the specified position. During interval [a,b], leg 2 is in motion and both leg 1 and the body remained at their previous locations, therefore the value of XPA[1] does not change. Since XPA[1] will become the foothold of leg 3 in the upcoming cycle, its value must be updated whenever the body is moving. This is observed in the figure throughout interval [b,c], as the value of XPA[1] decreases from 32 to 23, implying that the body has moved

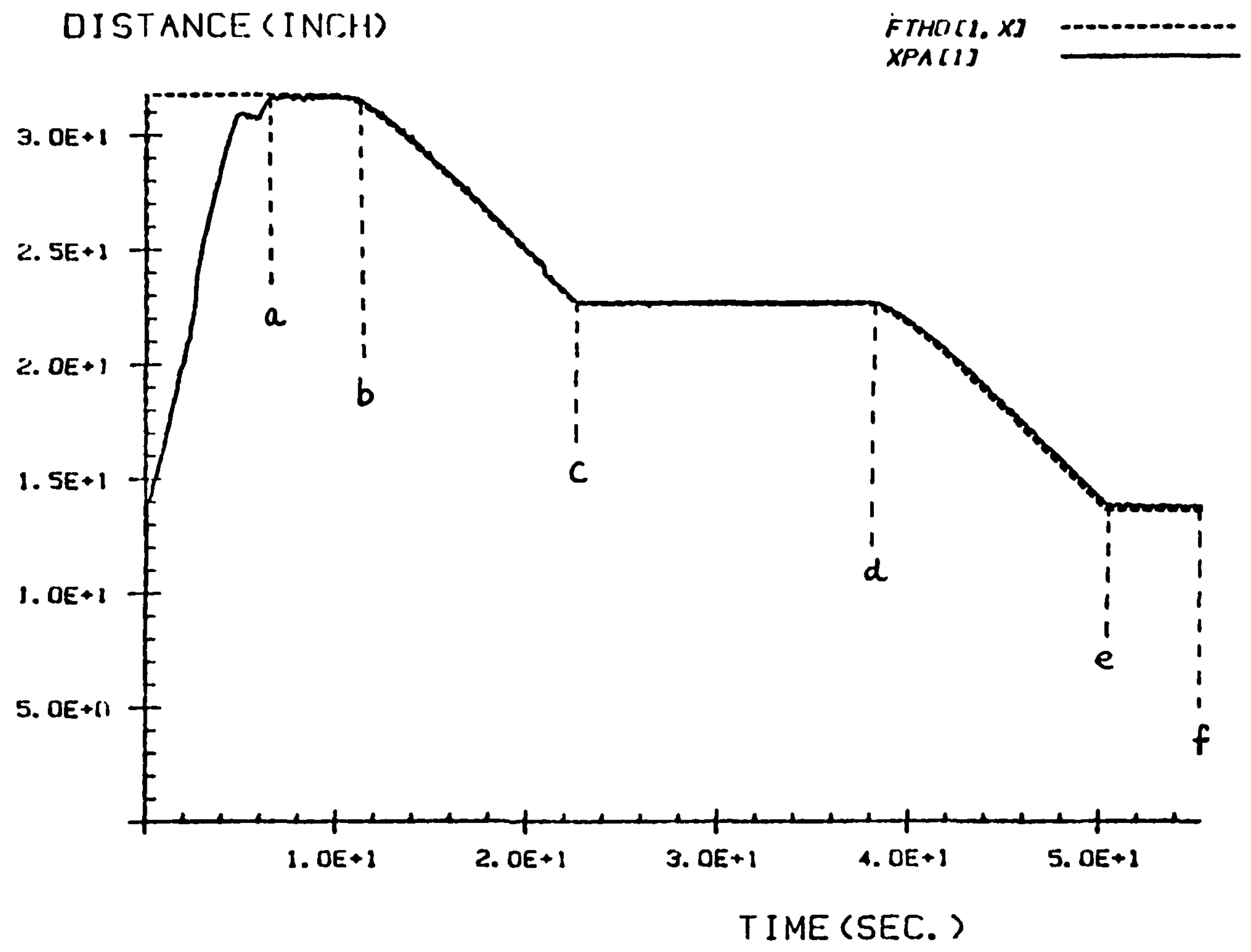


Figure 7.2 X-Coordinate of the Actual Foot Position of Leg 1, XPA[1], versus that of the Specified Foothold, FTHD[1,x].

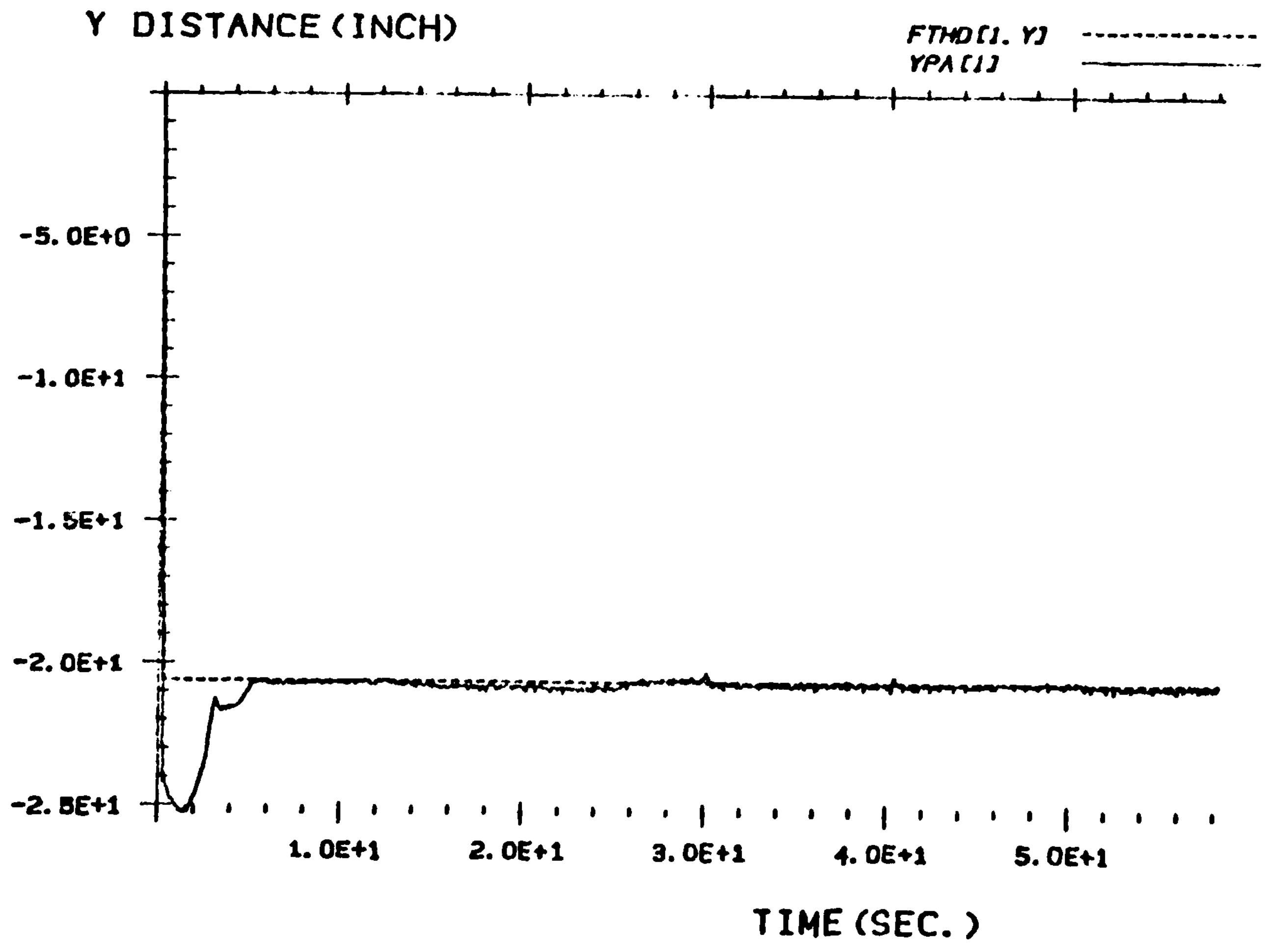


Figure 7.3 Y-Coordinate of the Actual Foot Position of Leg 1, YPA[1], versus that of the Specified Foothold, FTHD [1,Y].

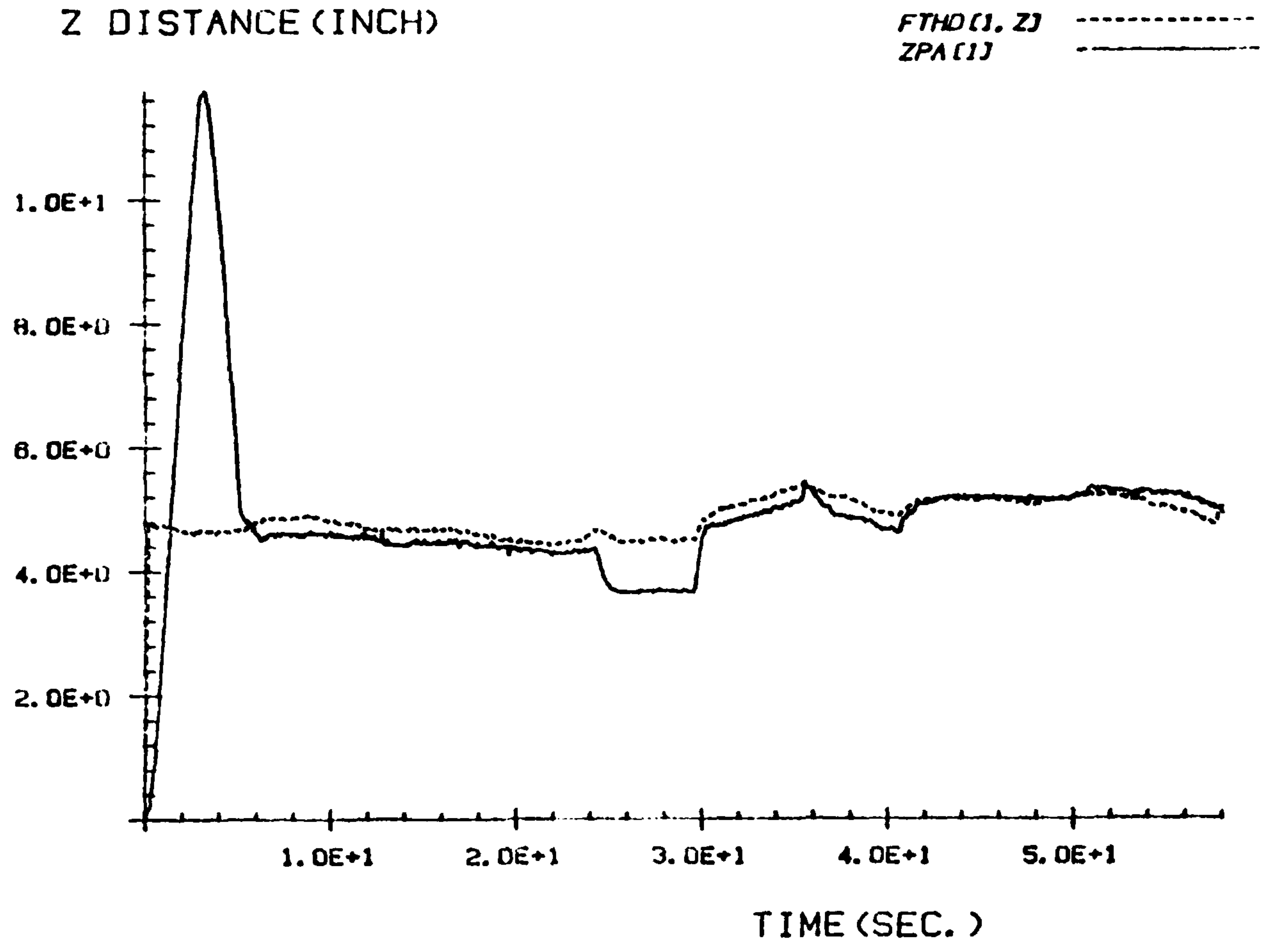


Figure 7.4 Z-Coordinate of the Actual Foot Position of Leg 1, ZPA[1], versus that of the Specified Foothold, FTHD[1,Z].

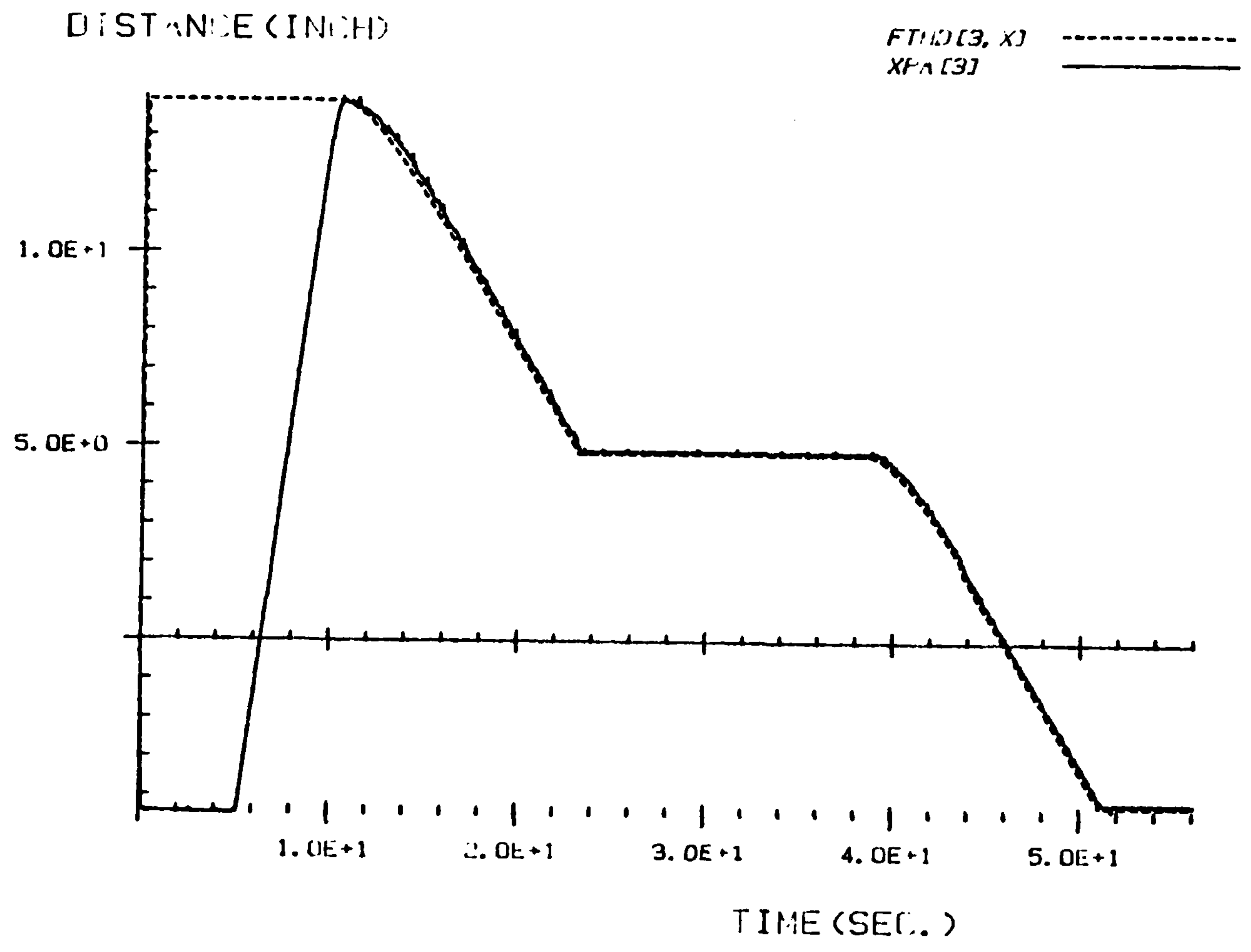


Figure 7.5 X-Coordinate of the Actual Foot Position of Leg 3, XPA[3], versus that of its Foothold, FTHD[3,X].

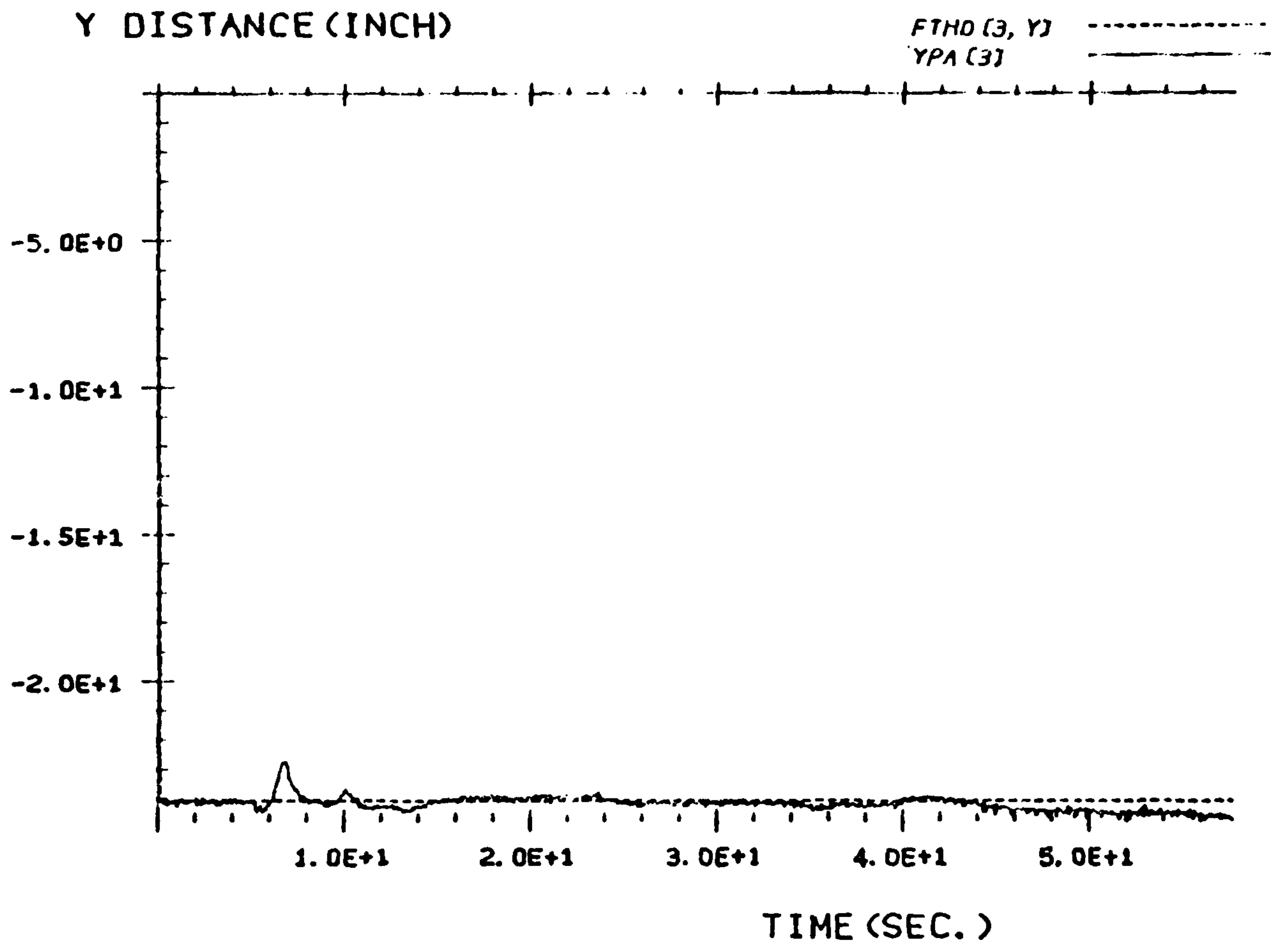


Figure 7.6 Y-Coordinate of the Actual Foot Position of Leg 3, YPA[3], versus that of its Foothold, FTHD[3,Y].

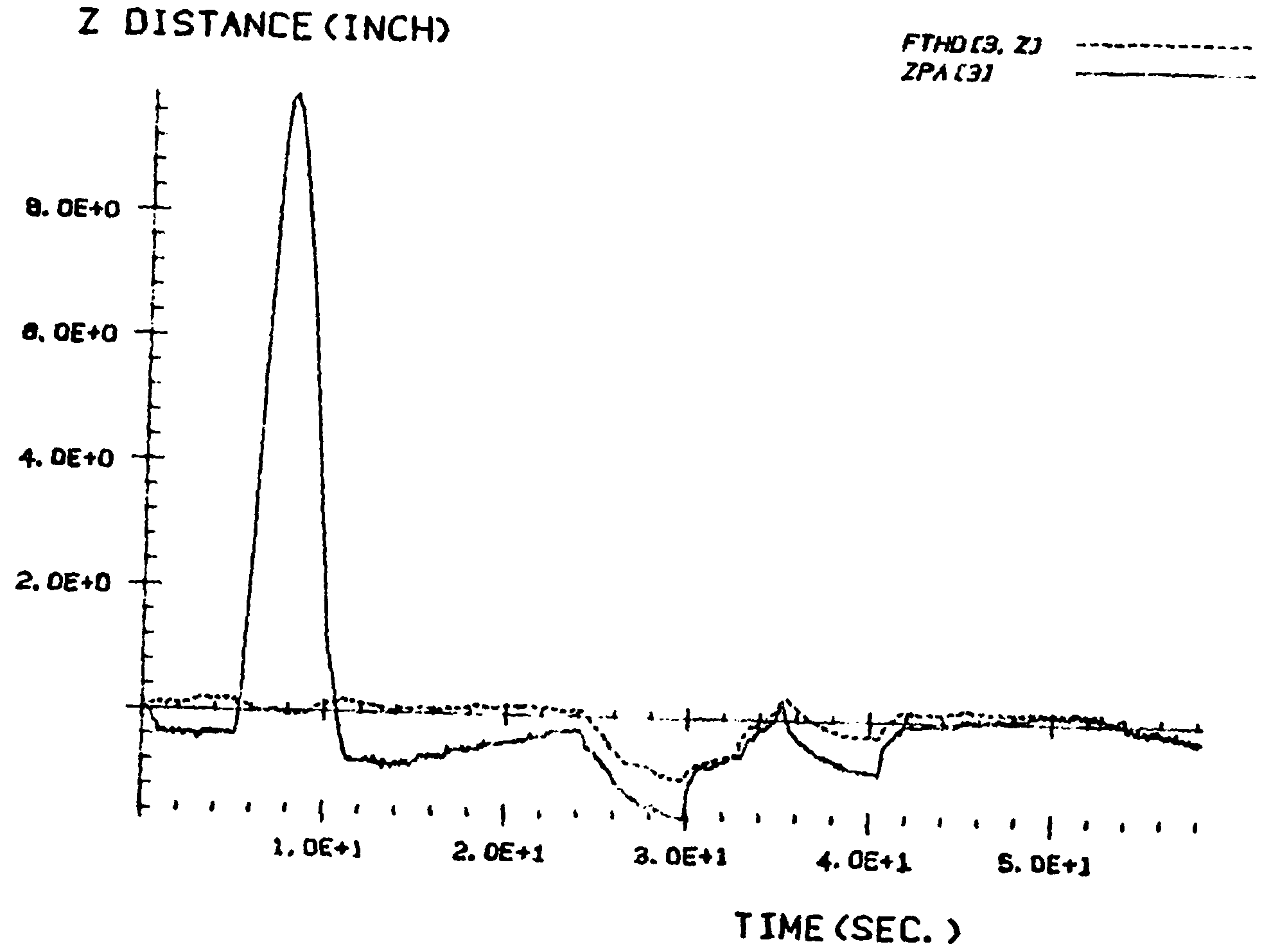


Figure 7.7 Z-Coordinate of the Actual Foot Position of Leg 3, ZPA[3], versus that of its Foothold, FTHD[3,Z].

forward for a distance of 9 inches. During [c,d], leg 5 moves, another foothold is assigned, and then leg 2 transfers toward it. The constant value of XPA[1] again indicates that neither the body nor legl moves in this period. The second body motion occurred between d and e, and XPA[1] is modified to a new value. Finally, leg 6 takes off in [e,f] and the whole cycle is completed. To summarize, Fig. 7.2 conveys the following information: first, leg 1 steps onto the assigned foothold as required, and second, the foothold is properly updated for the next locomotion cycle.

Fig. 7.3 concerns the Y component of leg 1 and its foothold. In this case, the Y coordinate of the assigned foothold is -20 inches and that of the initial position of leg 1 is -25. When moving in the air, YPA[1] approches gradually toward the prescribed value and finally meets it as it is placed down. Again, the agreement of the solid and the dashed lines implies that the motion of leg 1 in the Y direction complies with the requirement.

Fig. 7.4 illustrates the behavior of leg 1 in the Z direction. Originally, leg 1 is on the floor thus ZPA[1] has a initial value of 0 inch. The specified foothold lies on a wooden block which is about 4 inches high. Hence after leg 1 is placed down, its foot tip position is 4 inches above the floor.

The next three figures, Fig. 7.5 through Fig. 7.7, document the action of leg 3 during the cycle. They are parallel to those three which have just been described, and can be interpreted in exactly the same way. There is one important connection between the figures of leg 3 and those of leg 1. Since leg 3 must move to the location which is previously occupied by leg 1, the initial foot position of leg 1 should be consistent with the foothold position of leg 3 in all X, Y, and Z directions. Comparing Fig. 7.2 with Fig. 7.5, it can be seen that XPA[1] has an initial value of 14 inches, so does FTHD[3,x]. Similarly, YPA[1] in Fig. 7.3 has an initial value of -24 inches and FTHD[3,y] in Fig. 7.6 also has the same value, and ZPA[1] in Fig. 7.4 begins with a value of 0 inch which agrees with the value of FTHD[3,z] in Fig. 7.7.

Fig. 7.8 to Fig. 7.10 present a sequence of pictures taken during a FTL locomotion. In Fig. 7.8, leg 1 has just been placed down on the tiny bright spot which is the image of a laser beam. In Fig. 7.9, another foothold, the bright image on the floor, is specified for leg 2, which is moving toward it. Fig. 7.10 shows that leg 2 indeed steps onto the assigned foothold.

7.4 Summary

The performance of FTL operation depends on the accuracy of the vision system in locating the specified foothold, and the precision of the servo control loop in moving a leg to a commanded location. Also, coordinates of footholds should be updated accurately for usage in subsquent locomotion cycles.

The constructed vision system costs about $10,000. Its accuracy, in average, is about 1.69% of the distance vector between a testing point and the middle of the two cameras. It takes around 125 msec. to detect an image and to compute its 3-dimensional coordinates. Its operations are subject to the following restrictions: (1) only the first bright image is detected in a frame, (2) The image must lie

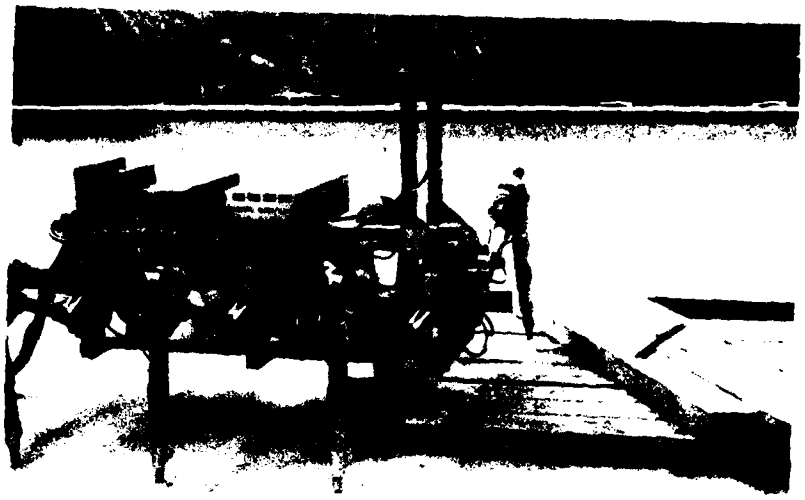


Figure 7.8 Picture Showing Leg 1 Stepping onto an Assigned Foothold.

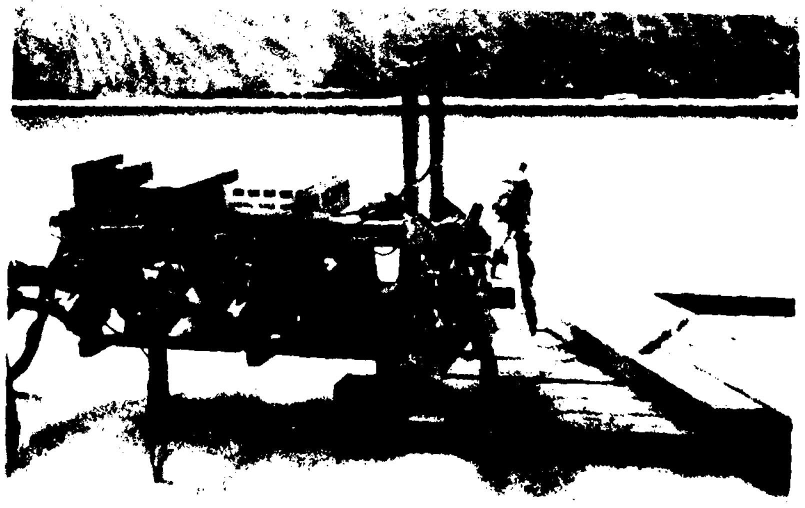


Figure 7.9 Picture Showing Leg 2 in the Process of Moving Toward an Assigned Foothold.

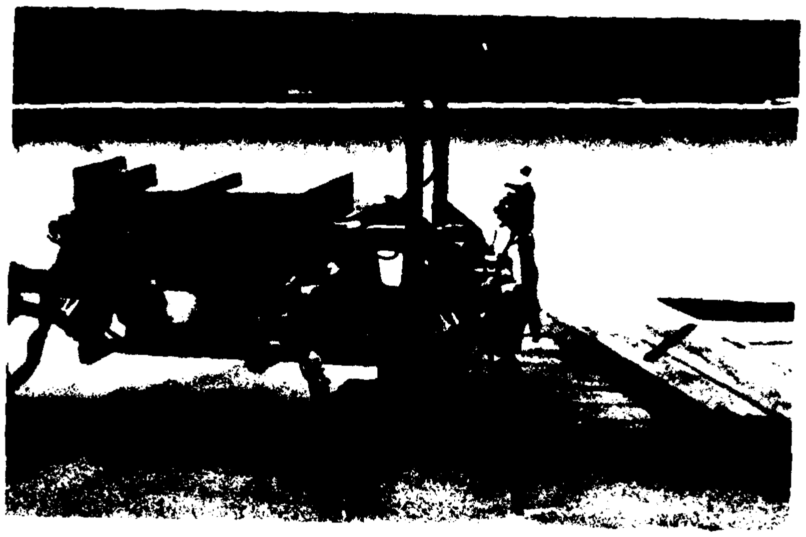
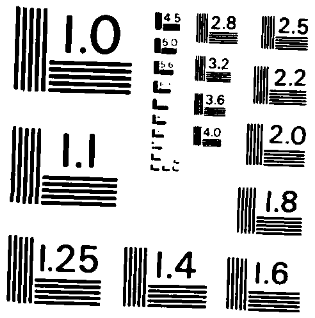


Figure 7.10 Picture Showing Leg 2 Stepping onto an Assigned Foothold.

inside the common field of view of the cameras, and (3) the laser beam must be aimed on the surface of an object which allows sufficient reflection.

Typical results obtainable from FTL operation were presented as sets of figures which depict the location of the assigned footholds and trajectories of legs. The results indicate that legs indeed steps onto the footholds, thus satisfying the requirement of FTL operation.


1.0
4.5
5.0
5.6
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

Chapter 8

SUMMARY AND CONCLUSION

8.1 *Research Contribution*

As an intermediate step toward the realization of a fully autonomous walking vehicle, in this research Follow-The-Leader operation was studied and successfully implemented. This achievement was made possible by first conceptualizing, then defining the problems, and finally pursuing appropriate solutions. Hardware and software needed to facilitate these solutions were then designed, constructed,, tested, and integrated. As a result, less decision-making effort is needed from an operator to assist the Hexapod walking over even/rough terrain than that in a joystick control mode, which implies a significant increase in the capability of the Hexapod. All in all, at the completion of this dissertation, the Hexapod was able to walk, step-by-step, over rough terrain under the guidance of an operator.

With regard to hardware, a binocular vision system based on two solid-state TV cameras was built. The components, such as cameras, lenses, laser unit, etc., were selected after careful study and comparisons so that the requirements regarding field of view, signal-to-noise ratio, installation convenience, etc., were all satisfied. A special-purpose interface circuit which served as the image coordinate decoder as well as the communication channel between cameras and a PDP-11/70 computer was designed and implemented. Cameras were calibrated to provide an accurate camera model. The fisheye lenses were included in this calibration to compensate distortion. The geometric relationship between cameras and the Hexapod were identified. A triangulation program was developed to reconstruct the three-dimensional coordinates of an observed light source. Consequently, a reliable vision system which is able to recognize specified footholds and locate their positions is now available to the Hexapod.

Before this work, Hexapod locomotion was limited to wave gaits based on considerations of stability margin, and the movements of all legs had to accommodate the motion of the center of the body. For FTL locomotion, however, the center of the body has to accommodate the motion of legs and wave gaits are not applicable to this case. Thus, a new method of coordinating stepping of successive legs so that only footholds which have been tested by preceding legs are used was discovered. Particularly, an algorithm was conceived to direct the center of the body toward a location which would provide an optimal stability margin. While a general theory which covers all classes of possible gaits and guarantees an optimal margin does not exit, it is believed that this algorithm can be used in locomotion with any kind of gait. As such, it makes feasible the automatic control of the Hexapod during climbing of large obstacles.

8.2 Research Extensions

As this work comes to an end, it becomes clear that, with additional work, the Hexapod can be made more versatile and its performance upgraded. With this in mind, a number of suggestions are given below.

(1) Terrain modeling. An automatic scanning system, which controls the laser beam and causes it to sweep over an area under the command of a computer, would enable the Hexapod to create a terrain map based on which terrain analysis could be carried out.

(2) Terrain analysis. A series of algorithms should be developed for the Hexapod to select good footholds from a terrain model. With this capability, the Hexapod would be able to choose a proper route and navigate itself through rough terrain.

The combination of (1) and (2) would result in an automatic navigation system for the Hexapod. The human operator could be eliminated and the vehicle would become fully automatic.

(3) Image processing and pattern recognition. Currently, only one image was detected for a frame. This means that most of the information contained in a picture was not used. To exploit the merits of a vision system to the greatest extent possible, an image processor could be added to the existing system, and scene analysis could be performed based on the intensity data. For example, the boundary of two objects may be mapped into edges in a picture. Information extracted from intensity data and that obtained from distance data may complement each other and provide a certain degree of redundancy.

(4) Real-time experiments using different types of gaits. Although the walking algorithms developed in this work were mainly for FTL operation, the restriction stems from the fact that the next foothold of middle/rear legs overlaps with the current foothold of front/middle legs, which is so because there is only one foothold available for each leg per locomotion cycle. If there are more than one foothold for every leg, then the gait sequence imposed in FTL locomotion can be lifted and motions with various gaits are ready to be tested.

In conclusion, with the successful implementation of FTL operation, the author is convinced that a fully autonomous legged vehicle is a reachable goal, provided additional instrumentation and algorithms are available. While new devices would provide the machine extra information about the external world, new algorithms should be developed to best utilize the available information. Taking the realization of FTL operation as an example, the binocular vision system enables the Hexapod to perform visual data acquisition, and based on its outcome, the coordinates of a specified foothold, the OSP concept is devised to use this piece of information to navigate the vehicle in the most beneficial way. The author also expects that, during the course of building such a vehicle, various problems related to the nature of motion of a biological system would be encountered, and the recognition as well as the resolving of these issues would be an important milestone toward the understanding of the behavior of biological systems.

REFERENCES

[1] McGhee, R.B., "Vehicular Legged Locomotion," in Advances in Automation and Robotics, Ed. by G.N. Saridis, Jai Press, Inc., 1983.

[2] Baldwin, W.C., and Miller, J.V., Multi-Legged Walker, Final Report, Space General Corporation, El Monte, CA, Jan., 1966.

[3] Mosher, R.S., "Exploring the Potential of a Quadruped," SAE Paper No. 690191, International Automotive Engineering Conference, Detroit, MI, January, 1969.

[4] McGhee, R.B., "Robot Locomotion," in Neural Control of Locomotion, Ed. by R.M. Herman, et al., Plenum Publishing Corp., New York, 1976, pp. 237-246.

[5] Orin, D.E., "Supervisory Control of a Multilegged Robot," Robotics Research, Vol. 1, No. 1, Spring, 1982, pp. 79-91.

[6] Laux, E.G.,"Automated Investment Casting Shelling Process," in Industrial Robot, Vol. 2, Ed. by W.R. Tanner, Robotics International of SME, Dearborn, MI., 1981.

[7] Oakland, M.R., "Automated Aluminum Die Casting," in Industrial Robot, Vol. 2, Ed. by W.R. Tanner, Robotics International of SME, Dearborn, MI., 1981.

[8] Tanner, W.R., Editor, Industrial Robot, Vol. 1, Robotics International of SME, Dearborn, MI., 1981.

[9] Nitzan, D., "Assessment of Robotic Sensors," Workship on the Research Needed to Advance the State of Knowledge in Robotics, Newport, RI, April, 1980.

[10] Ballard, D.H., and Brown, C.M., Computer Vision, Prentice-Hall Inc., New Jersey, 1982.

[11] Moravec, H.P, "The Stanford Cart and the CMU Rover," Report of the Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA., January, 1983.

[12] Lewis, R.A., and Johnston, A.R., "A Scanning Laser Rangefinder for a Robotic Vehicle, " Proc. 5th Joint Conf. Artificial Intelligence, Cambridge, MA, Aug., 1977, pp. 762-768.

[13] McNellis, T.J., Evaluation of a Laser Triangulation Ranging System for Mobile Robots, RPI Technical Report MP-80, Rensselaer Polytechnic Institute, Troy, New York, Aug., 1982.

[14] Dodd, G.G., and Rossol, L., Editors, Computer Vision and Sensor-Based Robots, Plenum Press, New York, 1979.

[15] Rosenfeld, A., "Rapporteur for Sensing Systems," Workshop On The Research Needed to Advance the State of Knowledge in Robotics, Newport, RI, April, 1980.

[16] Anon., VICOM Product Description, Vicom Systems Inc., San Jose, CA, November, 1982.

[17] Baird, M.L., "An Application of Computer Vision to Automate IC Chips Manufacture," Proc. Third Int. Joint Conf. Pattern Recognition, Nov., 1976, pp. 3-7.

[18] Jarvis, J.F., "Research Directions in Industrial Machine Vision: A Workshop Summary," Computer, Vol. 15, No. 12, Dec., 1982, pp. 55-61.

[19] Kashioka, S., et al., "A Transistor Wire-Bonding System Utilizing Multiple Local Pattern Matching Techniques," IEEE Trans. SMC, Vol. 6, No. 8, Aug., 1976, pp. 562-570.

[20] Anon., SELSPOT II System, Selcom Selective Electronic Inc., Valdese, NC, May, 1981.

[21] Danielsson, P.E., and Levialdi, S.,"Computer Architectures for Pictorial Information Systems," Computer, Vol. 14, No. 11, Nov., 1981.

[22] Preston, K., "Cellular Logic Computer for Pattern Recognition," Computer, Vol. 16, No. 1, Jan., 1983.

[23] Nudd, G.R. et al., "Implementation of Advanced Real-Time Image Understanding Algorithms," Proc. Image Understanding Workshop, SAI-80-895-WA, Washington, D.C., 1979.

[24] Gonzalez, R.C., and Safabakhah, R., "Computer Vision Techniques for Industrial Applications and Robot Control," Computer, Vol. 15, No. 12, Dec., 1982.

[25] Rossol, L., "Vision and Adaptive Robots in General Motors," The 1st Int. Conf. on Robot Vision and Sensory Control, U.K., April, 1981, pp. 277-288.

[26] Carlisle, B., et al., "The PUMA/VS-100 Robot Vision System," The 1st Int. Conf. on Robot Vision and Sensory Control, U.K., April, 1981, pp. 149-160.

[27] Makhlin, A.G., "Westinghouse Visual Inspection and Industrial Robot Control System," The 1st Int. Conf. on Robot and Sensory Control, U.K., April, 1981, pp. 35-46.

[28] Birk, J., et al., The Fifth Report on General Methods to Enable Robots with Vision to Acquire, Orient and Transport Workpieces, University of Rhode Island, Kingston, RI., August, 1979.

[29] Kelley, R., et al., "Acquiring Connecting Rod Castings Using A Robot with Vision and Sensors," The 1st Int. Conf. on Robot and Sensory Control, U.K., April, 1981, pp. 169-178.

[30] Chen, N., Visually Estimating Workpiece Pose in a Robot Hand Using Feature Points Method, Ph.D. dissertation, University of Rhode Island, Kingston, RI, August, 1979.

[31] Raphael, B., The Thinking Computer: Mind Inside Matter, W.H. Freeman Company, San Francisco, 1976.

[32] Nevatia, R., Machine Perception, Prentice-Hall, Inc., New Jersey, 1982.

[33] Shirai, Y., and Suwa, M., "Recognition of Polyhedrons with a Range Finder," Proc. 2nd Int. Joint Conf. Artificial Intelligence, London, England, Sep., 1971, pp. 80-87.

[34] Agin, G.J., and Binford, T.O., "Computer Description of Curved Objects," Proc. 3rd Int. Joint Conf. Artificial Intelligence, Standford, CA, Aug., 1973, pp. 629-640.

[35] Hart, P.E., et al., "Artificial Intelligence--Research and Applications," Annual Technical Report to ARPA, SRI, Menlo Park, CA, Contract DAHC04-72-C-008, Dec., 1972.

[36] Broerman, K.R., Development of a Proximity Sensor System for Control of Foot Altitude During Locomotion of a Hexapod Robot, M.S. thesis, The Ohio State University, Columbus, Ohio, June, 1983.

[37] Nitzan, D., Brain, A.E., and Duda, R.O., "The Measurement and Use of Registered Reflection and Range Data in Scene Analysis," Proc. IEEE, Vol. 65, No. 2, 1977, pp. 206-220.

[38] Ready, J.F., Industrial Applications of Lasers, Academic Press Inc., New York, 1978.

[39] Sun, S.S., A Theoretical Study of Gaits for Legged Locomotion Systems, Ph.D. dissertation, The Ohio State University, Columbus, Ohio, March, 1974.

[40] Orin, D.E., Interactive Control of a Six-Legged Vehicle with Optimization of Both Stability and Energy, Ph.D. dissertation, The Ohio State University, Columbus, Ohio, March, 1976.

[41] Jaswa, V.C., An Experimental Study of Real-Time Computer Control of a Hexapod Vehicle, Ph.D. dissertation, The Ohio State University, Columbus, Ohio, June, 1978.

[42] Buckett, J.R., Design of an On-Board Electronic Joint Control System for a Hexapod Vehicle, M.S. thesis, The Ohio State University, Columbus, Ohio, March, 1977.

[43] Chao, C.S., Real-Time Multiprocessor Control of a Hexapod Vehicle, Ph.D. dissertation, The Ohio State University, Columbus, Ohio, August, 1979.

[44] Briggs, R.L., A Real-Time Digital System for Control of a Hexapod Vehicle Utilizing Force Feedback, Ph.D. dissertation, The Ohio State University, Columbus, Ohio, March, 1979.

[45] Klein, C.A., and Wahawisan, W., "Use of a Multiprocessor for Control of a Robotic System," International Journal of Robotics Research, Vol. 1, No. 2, Summer, 1982, pp. 45-59.

[46] Pugh, D.R., An Autopilot for a Terrain-Adaptive Hexapod Vehicle, M.S. thesis, The Ohio State University, Columbus, Ohio, Sep., 1982.

[47] Zuk, D.M., and Dell'eva M.L., Three-Dimensional Vision System for the Adaptive Suspension Vehicle, Final Report, Environmental Research Institute of Michigan, Ann Arbor, MI, January, 1983.

[48] Anon., General Electric TN2500 Solid State Video/Digital Camera Operating Manual, Optoelectronic Systems Operation, Electronic Systems Division, General Electric Company, Syracuse, NY, April, 1980.

[49] Anon., Complete Catalog of Optical Systems and Components, ORIEL Corporation, Stamford, CT., 1979.

[50] Anon., PDP11 Peripherals Handbook, Digital Equipment Corporation, 1976.

[51] Anon., RSX-11 M/M-PLUS I/O Drivers Reference Manual, Digital Equipment Corporation, 1979.

[52] Smith, W.J., Modern Optical Engineering, McGraw-Hill Book Company, 1966.

[53] Anderson, N., Numerical Methods, Prentice-Hall, Inc., New Jersey, 1974.

[54] Duda, R.O., and Hart, P.E., Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[55] Paul, R.P., Robot Manipulators: Mathematics, Programming, and Control, The MIT Press, Cambridge, MA, 1981.

[56] Wahawisan, W., A Multiprocessor System with Applications to Hexapod Vehicle Control, Ph.D. dissertation, The Ohio State University, Columbus, Ohio, Sep., 1981.

[57] Bessonov, A.P., and Umnov, N.V., "The Analysis of Gaits in Six-Legged Vehicles According to Their Static Stability," Proc. of CISM-IFTOMM Symposium on Theory and Practice of Robots and Manipulators, Udine, Italy, September, 1973.

[58] Klein, C.A., and Huang, C.H., "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators," IEEE Trans. on Systems, Man, and Cybernatics, Vol. 12, April, 1983, pp. 245-250.

[59] Huang, C.H., The Use of Pseudoinverse for Kinematically Redundant Manipulator Systems, M.S. thesis, The Ohio State University, Columbus, Ohio, April, 1981.

[60] Kirk, D.E., Optimal Control Theory, Prentice-Hall Inc., New Jersey, 1970.

[61] McGhee, R.B., and Sun, S.S., "On the Problem of Selecting a Gait for a Legged Vehicle," Proc. of VI IFAC Symposium on Automatic Control in Space, American SSR, USSR, August, 1974.

[62] Klein, C.A., and Briggs, R.L., "Use of Active Compliance in the Control of Legged Vehicle," IEEE Trans. on Systems, Man, and Cybernatics, Vol. SMC-10, No. 7, July, 1980, pp. 393-400.

[63] Miller, P.E., An Investigation of Boolean Image Neighborhood Transformations, Ph.D. dissertation, The Ohio State University, Columbus, OH, December, 1978.

[64] McGhee, R.B., "Small Motion Dynamic Analysis of ASV-84 Attitude, Altitude, and Force Control Loops," Technical Note No. 25, Digital Systems Lab., The Ohio State University, August, 1982.

[65] Klein, C.A., Olson, K.W., and Pugh, D.R., "Use of Force and Attitude Sensors for Locomotion of a Legged Vehicle Over Irregular Terrain," The International Journal of Robotics Research, Vol. No. 2, Summer, 1983.

[66] McGhee, R.B., and Pai, A.L., "An Approach to Computer Control for Legged Vehicles," Journal of Terramechanics, Vol. 11, No. 1, January, 1974.

[67] Fang, R.C., A Digital TV System for Detection of High Speed Human Motion, Ph.D. dissertation, The Ohio State University, Columbus, Ohio, June, 1981.

APPENDIX

VISION SOFTWARE AND THE FOLLOW-THE-LEADER WALKING PROGRAMS

```
(*$A-*)
          {  FILE:  GBLF37.PAS  }

/*************************************************************************/
/*                                                                       */
/* FUNCTION:    THIS FILE CONTAINS GLOBAL DECLARATIONS FOR               */
/*              FTL MOTION SOFTWARE. THE EXECUTABLE FILES WHICH          */
/*              SHARE THESE GLOBALS ARE:                                 */
/*                                                                       */
/*              WALK37.PAS       VISION.PAS       FOOT37.PAS             */
/*              INIT37.PAS       LINE37.PAS       SERV37.PAS             */
/*              DLNK37.PAS       LIBR37.PAS       POWR37.PAS             */
/*              TRION.PAS        CHECK.PAS                               */
/*************************************************************************/

CONST
        PI        = 3.14159;
        FSCALE    = -200.0;       /* FORCE SCALE FACTOR               */
        MAXSTROKE = 12.0;         /* MAXIMUM FOOT STROKE              */
        MOVE      = TRUE;         /* SWITCH TO ENABLE SERVOING        */
        NOMOVE    = FALSE;        /* SWITCH TO DISABLE SERVOING       */


TYPE
        ARRAY6  = ARRAY[1..6] OF REAL;
        ARRAY18 = ARRAY[0..17] OF REAL;
        MODETYPE = ( RANDOM, NEUTRAL, PREWALK,
                     CRUISE, SIDESTEP, TURN    );
        AXES = (X,Y,Z);
        VECTOR = ARRAY[AXES] OF REAL;
        MATRIX =ARRAY[1..6,AXES] OF REAL;
        HMATRIX=ARRAY[1..4,1..4] OF REAL;
        ACTIVITY = (LEGMOTION,BODYMOTION,FOOTHOLDFETCH);
        LEGSET= SET OF 1..6;

 VAR
        MIDSTX, MIDSTY,           /* MIDSTANCE COORDINATES            */
        RPHASE,                   /* RELATIVE LEG PHASES              */
        XFA,  YFA,  ZFA,          /* ACTUAL FOOT FORCES               */
        XFD,  YFD,  ZFD,          /* DESIRED FOOT FORCES              */
        XSTATE,YSTATE,ZSTATE,     /* FILTERED FOOT FORCES             */
        XPA,  YPA,  ZPA,          /* ACTUAL FOOT POSITIONS            */
        XPD,  YPD,  ZPD,          /* DESIRED FOOT POSITIONS           */
        XRD,  YRD,  ZRD,          /* DESIRED FOOT RATES               */
        ZPTERM,                   /* Z POS. TERM FOR ATT. CONTROL     */
        ZZERO,                    /* Z COMPONENT OF ZEROFORCE         */
        RSUPPORT                  /* SUPPORT PHASE INDICATORS         */

                : ARRAY6;
```

```
ZEROFORCE,                  /* TRUE FORCE OFFSETS              */
FORCE                       /* TRUE FORCE                      */
        : ARRAY18;

FZERO                       /* FLAGS FOR FORCE ZEROING         */

        : ARRAY[1..6] OF BOOLEAN;

ATTITUDE,                   /* SWITCH FOR ATTITUDE CONTROL     */
COMPLIANCE,                 /* INDIRECT COMPLIANCE SWITCH      */
OPTIMIZATION,               /* SWITCH FOR OPTIMAL FORCE        */
PASS1,                      /* SWITCH TO INITIALIZE FILTER     */
SAVE,                       /* SWITCH FOR DATA AQUISITION      */
SPRING                      /* SWITCH FOR ACTIVE COMPLIANCE    */

        : BOOLEAN;

COMMAND                     /* OPERATOR INPUT COMMAND          */

        : CHAR;

DATACLOCK,                  /* CLOCK AT ACQUISITION START      */
LASTCLOCK,                  /* STORAGE FOR CLOCK BUFFER        */
TOTALCLOCK,                 /* CLOCK TICK ACCUMULATOR          */
TTICK,                      /* DATA STORAGE COUNTER            */
XP1,YP1,XP2,YP2,
STOP,SBG,AG
        : INTEGER;

MODE                        /* HEXAPOD OPERATING MODE          */

        : MODETYPE;

BETA,                       /* LEG DUTY FACTOR                 */
BPHASE,
DPSI,                       /* FILTERED TURN RATE COMMAND      */
DT,                         /* DELTA TIME (SEC)                */
DXB,
DYB,
DZB,
FOOTLIFT,                   /* FOOT LIFTING HEIGHT             */
LIFTTIME,
MIDSTZ,                     /* MIDSTANCE Z COORDINATE          */
NVELX, NVELY,               /* OPERATOR VELOCITY COMMANDS      */
NDPSI,                      /* OPERATOR TURN RATE COMMAND      */
PERIOD,                     /* PERIOD OF KINEMATIC CYCLE       */
PHASE,                      /* KINEMATIC CYCLE PHASE           */
PSIC,
RADIUS,                     /* RADIUS FROM CG. TO MIDSTANCE    */
TPHASE,
SPERIOD,                    /* SUPPORT PERIOD (SEC)            */
VELX, VELY,                 /* FILTERED VELOCITY COMMANDS      */
VELMAX,                     /* MAX FOOT VELOCITY COMPONENT     */
XT,YT,ZT,XE,YE,ZE,ZFS
        : REAL;

SUPPORT                     /* SET OF LEGS IN SUPPORT PHASE    */

        : LEGSET;
TRISET  : ARRAY[1..20] OF LEGSET;
FTHD,PTFTHD : MATRIX;
TMATRIX:HMATRIX;
ACTION : ACTIVITY;
VTR12,VTR23 : VECTOR;
```

```
(*$A-*)
                { FILE: WALK37.PAS }

   /*****************************************************/
   /* PROGRAMMER : SHENJ-JEN TSAI                      */
   /* DATE : APRIL 20, 1983                            */
   /* FUNCTION : THIS PROGRAM DIRECTS HEXAPOD TO       */
   /*            PERFORM FOLLOE-THE-LEADER MOTION      */
   /*            WITH GAIT SEQUENCE 1-3-5-2-4-6.       */
   /*                                                 */
   /* PROCEDURES CALLED : FOOT37, INIT37, LIBR37,     */
   /*                     SERV37, DLNK37, LINE37,     */
   /*                     POWR37, VISION, TRION,      */
   /*                     ROOT4,  CHECK,              */
   /*                                                 */
   /* (NOTE : INIT37, LIBR37, SERV37, DLNK37, LINE37, */
   /*         AND POER37 ARE EXACTLT THE SAME AS THOSE */
   /*         USED IN VERSION 3.0 SOFTWARE. THEY ARE   */
   /*         DOCUMENTED IN PUGH'S THESIS. )           */
   /*                                                 */
   /* USER GUIDE :  >PAS WALK37=GBLF37,WALK37         */
   /*               >MAC WALK37=WALK37                */
   /*               >TKB @WALK37.CMD                  */
   /*               >RUN WALK37                       */
   /*                                                 */
   /*****************************************************/

VAR
   COMPONENT:AXES;
   NEXT,CG,CGMOVE,OLDFT1,OLDFT2,NEWFT1,
   NEWFT2,P1,P2,LTGC,PTGC,NTGC :VECTOR;
   XTEMP,YTEMP,ZTEMP,ERR,XG,YG,ZG,SAI,TH1,TH2,
   XSTRIDE,YSTRIDE,XABSTK,YABSTK,
   TTIME,DVELX,DVELY,DDPSI,PITCH,ROLL,
   ANGCG,AUXANG,FGAIN,VMAX,STROKE,PTIME,LTIME,
   LANG,RANG,ANG,MAXANG,A,B,C,TURANG,RESANG,
   DXCG,DYCG,DANGCG,TANGCG:REAL;
   J,LEGLIFT,ID,I,NT:INTEGER;
   SWITCH:CHAR;
   SAFE,CONTACT:BOOLEAN;
   HIPX,OLDXPA,OLDYPA,OLDZPA:ARRAY6;
   TOPBLOCK:ARRAY18;

LABEL  20,30;

FUNCTION  ATAN2(Y, X : REAL) : REAL; EXTERNAL;

FUNCTION  ASIN(VALUE : REAL) : REAL; EXTERNAL;

FUNCTION  SIGN(X:REAL):REAL; EXTERNAL;

PROCEDURE  CLOCKINIT; EXTERNAL;
PROCEDURE  VISION; EXTERNAL;

PROCEDURE  NORMALIZE; EXTERNAL;

PROCEDURE  RBADC(NOCHAN,FRSTCH:INTEGER; SCALE:REAL;
                 VAR  INDATA:ARRAY18); EXTERNAL;

PROCEDURE  CHECK(XG,YG,ZG :REAL; VAR  SAI,TH1,TH2 :REAL);
           EXTERNAL;

PROCEDURE  INITIALIZE;  EXTERNAL;

PROCEDURE  HALT;  EXTERNAL;

PROCEDURE  FOOTPATH;  EXTERNAL;

PROCEDURE  TESTON;  EXTERNAL;

PROCEDURE  TURNON;  EXTERNAL;

PROCEDURE  TURNOF;  EXTERNAL;

PROCEDURE  JSERVO(MOVE: BOOLEAN ); EXTERNAL;

PROCEDURE  TURNANGLE;  { COMPUTE THE TURNING ANGLE }

   BEGIN
     VTR12[X]:=PTGC[X]-LTGC[X];
     VTR12[Y]:=PTGC[Y]-LTGC[Y];
     VTR23[X]:=NTGC[X]-PTGC[X];
     VTR23[Y]:=NTGC[Y]-PTGC[Y];
     A:=SQRT(VTR12[X]*VTR12[X]+VTR12[Y]*VTR12[Y]);
     B:=SQRT(VTR23[X]*VTR23[X]+VTR23[Y]*VTR23[Y]);
     C:=(VTR12[X]*VTR23[X]+VTR12[Y]*VTR23[Y])/(A*B);
     IF ABS(C) > 0.9999 THEN ANG:=0.
     ELSE BEGIN
           IF VTR23[Y] > VTR12[Y] THEN A:=SQRT(1.-C*C)
           ELSE A:=-SQRT(1.-C*C);
           ANG:=ATAN2(A,C);
           IF ABS(ANG) >= (3.14159/2.)
            THEN BEGIN
                   ANG:=ANG-3.14159;
                   IF ABS(ANG) >= 3.14159 THEN ANG:=ANG+2.*3.14159
                   ELSE;
                 END
            ELSE;
           END;
          ANG:=ANG+RESANG;
          IF ABS(ANG) > ABS(MAXANG) THEN TURANG:=MAXANG*ANG/ABS(ANG)
                                    ELSE TURANG:=ANG;
          NDPSI:=TURANG/PERIOD;
          ANG:=TURANG*180./3.14159;
   END;  { OF TURNANGLE }

PROCEDURE  SYNTHMOTION; { COMPUTE X AND Y VELOCITIES }

    BEGIN
          DT:=0.;
          XSTRIDE:=1.2*(NTGC[X]-PTGC[X]);
          YSTRIDE:=1.2*(NTGC[Y]-PTGC[Y]);
```

```
        XABSTK:=ABS(XSTRIDE);  YABSTK:=ABS(YSTRIDE);
        IF XABSTK >= YABSTK THEN
           BEGIN
             VMAX:=XABSTK/PERIOD;
             IF XABSTK >= 0.001 THEN
               BEGIN
                 NVELX:=VMAX*XSTRIDE/XABSTK;
                 NVELY:=VMAX*YSTRIDE/XABSTK;
               END
             ELSE
               BEGIN
                 NVELX:=0.;  NVELY:=0.;
               END;
           END
        ELSE
          BEGIN
            VMAX:=YABSTK/PERIOD;
            IF YABSTK >= 0.001 THEN
              BEGIN
                NVELX:=VMAX*XSTRIDE/YABSTK;
                NVELY:=VMAX*YSTRIDE/YABSTK;
              END

            ELSE
              BEGIN
                NVELX:=0.;  NVELY:=0.;
              END;
          END;
    END;  { OF SYNTHMOTION }

PROCEDURE  NEWDATA; { FETCH THE NEXT FOOTHOLD }
VAR   XD,YD,ZD:REAL;

BEGIN
VISION;
NEXT[X]:=XE*100./2.54+22.75+1.5;
NEXT[Y]:=YE*100./2.54;
NEXT[Z]:=ZE*100./2.54-10.5;
IF NEXT[Y] >= 0. THEN { FOOTHOLD IS FOR LEG 2 }
                  BEGIN
                    FTHD[2,X]:=NEXT[X]-0.25;
                    FTHD[2,Y]:=NEXT[Y]+0.4;
                    FTHD[2,Z]:=NEXT[Z];
                    ID:=2; LEGLIFT:=2;
                    NTGC[X]:=(XPA[1]+XPA[3]+XPA[5]+FTHD[2,X]
                              +FTHD[4,X]+FTHD[6,X])/6.;
                    NTGC[Y]:=(YPA[1]+YPA[3]+YPA[5]+FTHD[2,Y]
                              +FTHD[4,Y]+FTHD[6,Y])/6.;
                  END
                ELSE  { FOOTHOLD IS FOR LEG 1 }
                  BEGIN
                    FTHD[1,X]:=NEXT[X];
                    FTHD[1,Y]:=NEXT[Y];
                    FTHD[1,Z]:=NEXT[Z];
                    ID:=1; LEGLIFT:=1;
                    NTGC[X]:=(XPA[2]+XPA[4]+XPA[6]+FTHD[1,X]
                              +FTHD[3,X]+FTHD[5,X])/6.;
                    NTGC[Y]:=(YPA[2]+YPA[4]+YPA[6]+FTHD[1,Y]
                              +FTHD[3,Y]+FTHD[5,Y])/6.;
                  END;
```

```
ACTION:=LEGMOTION;
SUPPORT:=SUPPORT-[LEGLIFT];
END;

PROCEDURE SAFECK; { SIMULATE THE MOTION AND SEE IF ANY LEG
                    WILL BE OUT OF THE KINEMATIC MARGINS }

CONST YHIP=8.562;
VAR ICK:INTEGER;
    DXSM,DYSM,B,C:REAL;
    FTHDSM:MATRIX;

BEGIN
   SAFE:=TRUE; SAI:=0.; TH1:=0.; TH2:=0.;

{ CHECK IF LEG 3(4) WILL BE OUT OF MARGIN }

   ICK:=LEGLIFT+2;
   XG:=FTHD[ICK,X]-HIPX[ICK];
   YG:=ABS(FTHD[ICK,Y])-YHIP;
   ZG:=FTHD[ICK,Z];
   CHECK(XG,YG,ZG,SAI,TH1,TH2);
   SAI:=ABS(SAI);  TH1:=ABS(TH1);  TH2:=ABS(TH2);
   IF (SAI > 75.) OR (TH1 > 75.) OR (TH2 > 75.)
   THEN
     BEGIN
        WRITELN(' LEG ',ICK:1,' IS OUT OF MARGIN #1 ! ');
        WRITELN('SAI=',SAI:8:4,'  TH1=',TH1:8:4,'  TH2=',
                 TH2:8:4);
        SAFE:=FALSE;
     END
   ELSE;

{ CHECK IF ANY LEG WOULD BE OUT OF MARGIN IF THE BODY MOVES
  TO ITS DESTINATION }

   B:=SIN(TURANG);  C:=COS(TURANG);
   DXSM:=XSTRIDE/1.2;  DYSM:=YSTRIDE/1.2;
   FOR I:=1 TO 6 DO
     BEGIN
        IF ((ID=1) AND ((I=1) OR (I=3))) OR ((ID=2)
            AND ((I=2) OR (I=4)))
        THEN
        BEGIN
        FTHDSM[I,X]:=(FTHD[I,X]-DXSM)*C+(FTHD[I,Y]-DYSM)*B;
        FTHDSM[I,Y]:=-(FTHD[I,X]-DXSM)*B+(FTHD[I,Y]-DYSM)*C;
        XG:=FTHDSM[I,X]-HIPX[I];
        YG:=ABS(FTHDSM[I,Y])-YHIP;
        ZG:=FTHD[I,Z];
        END
        ELSE
        BEGIN
        XG:=(PTFTHD[I,X]-DXSM)*C+(PTFTHD[I,Y]-DYSM)*B
             -HIPX[I];
        YG:=-(PTFTHD[I,X]-DXSM)*B+(PTFTHD[I,Y]-DYSM)*C;
        YG:=ABS(YG)-YHIP;
        ZG:=PTFTHD[I,Z];
        END;
        CHECK(XG,YG,ZG,SAI,TH1,TH2);
        SAI:=ABS(SAI);  TH1:=ABS(TH1);  TH2:=ABS(TH2);
```

```
          IF (SAI > 75.) OR (TH1 > 75.) OR (TH2 > 75.)
          THEN
            BEGIN
              WRITELN(' LEG ',I:1,' IS OUT OF MARGIN #2 ! ');
              WRITELN('SAI=',SAI:8:4,'  TH1=',TH1:8:4,'  TH2=',
                          TH2:8:4);
              SAFE:=FALSE;
            END
          ELSE;
      END; { OF DO }

{ CHECK IF LEG 5(6) WILL BE OUT OF MARGIN AFTER THE BODY
  MOVES AND AFTER THE LEG TRANSFERS }

   ICK:=LEGLIFT+4;
   FTHDSM[ICK,X]:=(FTHD[ICK,X]-DXSM)*C+(FTHD[ICK,Y]-DYSM)*B;
   FTHDSM[ICK,Y]:=-(FTHD[ICK,X]-DXSM)*B+(FTHD[ICK,Y]-DYSM)*C;
   XG:=FTHDSM[ICK,X]-HIPX[ICK];
   YG:=ABS(FTHDSM[ICK,Y])-YHIP;
   ZG:=FTHD[ICK,Z];
   CHECK(XG,YG,ZG,SAI,TH1,TH2);
   SAI:=ABS(SAI);  TH1:=ABS(TH1);  TH2:=ABS(TH2);
   IF (SAI > 75.) OR (TH1 > 75.) OR (TH2 > 75.)
   THEN
     BEGIN
        WRITELN(' LEG ',ICK:1,' IS OUT OF MARGIN #3 ! ');
        WRITELN('SAI=',SAI:8:4,'  TH1=',TH1:8:4,'  TH2=',
                TH2:8:4);
        SAFE:=FALSE;
     END
   ELSE;
END; { OF SAFECK }

PROCEDURE FOOTHEIGHT; { COMPUTE FOOTLIFT OF A TRANSFER LEG }
VAR B,C:REAL;
BEGIN
   IF PTFTHD[LEGLIFT,Z] > FTHD[LEGLIFT,Z]
   THEN B:=FTHD[LEGLIFT,Z]
   ELSE B:=PTFTHD[LEGLIFT,Z];
   C:=B-5.;
   IF C > 10. THEN FOOTLIFT:=B-10.
   ELSE FOOTLIFT:=5.8;
END; { OF FOOTHEIGHT }


BEGIN   { BEGIN OF THE MAIN PROGRAM }

CLOCKINIT;  { START PROGRAMMABLE CLOCK }


        MIDSTX[1]:=22.75;         MIDSTX[2]:=22.75;
        MIDSTX[3]:=0.;            MIDSTX[4]:=0.;
        MIDSTX[5]:=-22.75;        MIDSTX[6]:=-22.75;
        MIDSTY[1]:=-24.;          MIDSTY[2]:=24.;
        MIDSTY[3]:=-24.;          MIDSTY[4]:=24.;
        MIDSTY[5]:=-24.;          MIDSTY[6]:=24.;
        MIDSTZ:=20.;              BETA:=5/6;
        HIPX[1]:=22.75;           HIPX[2]:=22.75;
        HIPX[3]:=0.;              HIPX[4]:=0.;
        HIPX[5]:=-22.75;          HIPX[6]:=-22.75;
  FGAIN:=0.5;                FOOTLIFT:=10.;
  ZF5:=MIDSTZ;

  { INITIALIZE [illegible]N-COMPLIANT TRIPOD SETS IN ORDER
    OF PREFERENCE }

  TRISET[1] := [1,4,5];   TRISET[2] := [2,3,6];

  TESTON;
  TURNOF;
  WRITELN(' TURN ACTIVE COMPLIANCE ON ? Y/N ');
  READLN(SWITCH);
  IF SWITCH = 'Y' THEN COMPLIANCE:= TRUE
  ELSE COMPLIANCE:= FALSE;
  WRITELN(' TURN ATTITUDE CONTROL ON ? Y/N ');
  READLN(SWITCH);
  IF SWITCH = 'Y' THEN ATTITUDE:=TRUE
  ELSE ATTITUDE:=FALSE;
  WRITELN(' TURN FORCE OPTIMIZATION ON ? Y/N ');
  READLN(SWITCH);
  IF SWITCH= 'Y' THEN OPTIMIZATION:=TRUE
  ELSE OPTIMIZATION:=FALSE;
  RBADC(5,58,1,TOPBLOCK);
  PITCH:=TOPBLOCK[7]*(-0.7854);
  ROLL:=TOPBLOCK[8]*(-0.7854);
  WRITELN('  PITCH=',PITCH:8:4,'  ROLL=',ROLL:8:4);
  WRITELN(' INPUT THE VALUE OF MAX. TURANG, IN DEGREE ');
  READLN(MAXANG);
  MAXANG:=MAXANG*3.14159/180.;
  PASS1:=FALSE;
  NORMALIZE;
  INITIALIZE;

{ MOVE HEXAPOD INTO THE INITIAL POSTURE FOR FTL MOTION }

  XPD[1]:=MIDSTX[1]-9.;
  XPD[2]:=MIDSTX[2];
  XPD[3]:=MIDSTX[3]-4.5;
  XPD[4]:=MIDSTX[4]+4.5;
  XPD[5]:=MIDSTX[5];
  XPD[6]:=MIDSTX[6]+9.;

  FOR I:= 1 TO 6 DO
     BEGIN
        YPD[I]:=MIDSTY[I];  ZPD[I]:=MIDSTZ;
        PTFTHD[I,X]:=XPD[I];
        PTFTHD[I,Y]:=YPD[I];
        PTFTHD[I,Z]:=ZPD[I];
     END;
  ERR:=1000.;
  TURNON;
  WHILE ERR > 1. DO
     BEGIN
          ERR:=0.;
          JSERVO(MOVE);
          FOR I:=1 TO 6 DO
             BEGIN
              ERR:=ERR+ABS(XPA[I]-XPD[I]);
             END;
     END;
```

```
 HALT;
 SUPPORT:=[1,2,3,4,5,6];

{ ASSIGN FOOTHOLDS FOR MIDDLE AND REAR LEGS }

 FOR I:=3 TO 6 DO
   BEGIN
     FTHD[I,X]:=XPA[I-2];
     FTHD[I,Y]:=YPA[I-2];
     FTHD[I,Z]:=ZPA[I-2];
   END;
 PTGC[X]:=0.;  PTGC[Y]:=0.;  PTGC[Z]:=MIDSTZ;
 LTGC[X]:=-10.;  LTGC[Y]:=0.;
 VELX:=0.;  VELY:=0.; DPSI:=0.;
 PERIOD:=11.;

{ FETCH THE NEXT FOOTHOLD FOR A FRONT FOOT,
  DETERMINE THE MOTION OF THE BODY, CHECK
  FOR KINEMATIC MARGINS }

 SAFE:=FALSE;
 WHILE NOT(SAFE) DO
   BEGIN
         NEWDATA;
         FOOTHEIGHT;
         TURNANGLE;
         SYNTHMOTION;
         SAFECK;
   END;
 TTIME:=0.;
 CGMOVE[X]:=0.;  CGMOVE[Y]:=0.;  CGMOVE[Z]:=0.;
 LTIME:=TIME;
 RESANG:=0.;
 LIFTTIME:=6.;
 TPHASE:=0.;  BPHASE:=0.;
 FOR COMPONENT:= X TO Z DO
   BEGIN
     P1[COMPONENT]:=FTHD[5,COMPONENT];
     P2[COMPONENT]:=FTHD[6,COMPONENT];
   END;
 OLDFT1:=P1;
 OLDFT2:=P2;
20:  CG[X]:=0.;  CG[Y]:=0.;  CG[Z]:=0.;
 ANGCG:=0.;
 TURNON;
 SPRING:=COMPLIANCE;

{ LOOP FOR A CYCLE OF FTL MOTION }

 WHILE NOT(ACTION=FOOTHOLDFETCH) DO
   BEGIN
    CASE ACTION OF
      BODYMOTION: BEGIN
       IF BPHASE <= 1. THEN { KEEP ON MOVING }
        BEGIN
         BPHASE:=BPHASE+DT/PERIOD;
         DVELX:= FGAIN*(NVELX - VELX);
         DVELY:= FGAIN*(NVELY - VELY);
         DDPSI:= FGAIN*(NDPSI - DPSI);
         VELX:= DVELX*DT+VELX;
    VELY:= DVELY*DT+VELY;
    DPSI:= DDPSI*DT+DPSI;
  END
 ELSE  { BODY STOP MOVING }
    BEGIN
      NVELX:=0.;  NVELY:=0.;  NDPSI:=0.;
      VELX:=0.;   VELY:=0.;  DPSI:=0.;
      DT:=0.;
      FOR I:= 1 TO 6 DO
      BEGIN
        PTFTHD[I,Z]:=ZPA[I];
      END;
      BPHASE:=0.;
      ACTION:=LEGMOTION;
      LEGLIFT:=LEGLIFT+2;
      FOOTHEIGHT;
      SUPPORT:=SUPPORT-[LEGLIFT];
    END;
  END;  { OF BODYMOTION }

 LEGMOTION:  BEGIN
               TPHASE:=TPHASE+DT/LIFTTIME;
              IF (TPHASE >= 1.) AND ( CONTACT ) THEN
                 BEGIN
                   TPHASE:=0.;  CONTACT:=FALSE;
                   PTFTHD[LEGLIFT,X]:=XPA[LEGLIFT];
                   PTFTHD[LEGLIFT,Y]:=YPA[LEGLIFT];
                   PTFTHD[LEGLIFT,Z]:=ZPA[LEGLIFT];
                   SUPPORT:=SUPPORT+[LEGLIFT];
                   IF (LEGLIFT=3) OR ( LEGLIFT=4)
                   THEN
                        BEGIN
                                ACTION:=BODYMOTION;

                        END
                   ELSE
                        IF (LEGLIFT=1) OR (LEGLIFT=2)
                        THEN
                          BEGIN
                            LEGLIFT:=LEGLIFT+2;
                            FOOTHEIGHT;
                            SUPPORT:=SUPPORT-[LEGLIFT];
                          END
                        ELSE
                           BEGIN

                            ACTION:=FOOTHOLDFETCH;
                            HALT;
                            SPRING:=COMPLIANCE;
                            GOTO 30;
                           END;
                 END
                ELSE;
         END;  { OF LEGMOTION }
     END;  { OF CASE ACTION }

JSERVO(NOMOVE);
FOR I:=1 TO 6 DO
   BEGIN
     OLDXPA[I]:=XPA[I];
```

```
               OLDYPA[I]:=YPA[I];
               OLDZPA[I]:=ZPA[I];
             END;
          FOOTPATH;
          JSERVO(NOMOVE);
          NT:=0;  DZB:=0.; ZF5:=0.;
          FOR I:=1 TO 6 DO
               BEGIN
                 IF ( I IN SUPPORT ) THEN
                     BEGIN
                          NT:=NT+1;
                          DZB:=DZB+OLDZPA[I]-ZPA[I];
                          ZF5:=ZF5+ZPA[I];
                     END   ELSE;
                END;

      { CHECK IF THE LEG MAKES CONTACT WITH GROUND }

          IF (TPHASE > 0.5) AND (ZFA[LEGLIFT] > 15.)
          THEN CONTACT:=TRUE
          ELSE CONTACT:=FALSE;
          DZB:=DZB/NT;
          ZF5:=ZF5/NT;

      { UPDATE THE FOOTHOLDS IN BODY CS. AS HEXAPOD IS
        MOVING. }

          B:=SIN(PSIC);   C:=COS(PSIC);
          FOR I:=1 TO 6 DO
            BEGIN
                XTEMP:=FTHD[I,X];  YTEMP:=FTHD[I,Y];
                ZTEMP:=FTHD[I,Z];
                FTHD[I,X]:=(XTEMP-DXB)*C+(YTEMP-DYB)*B;
                FTHD[I,Y]:=-(XTEMP-DXB)*B+(YTEMP-DYB)*C;
                FTHD[I,Z]:=ZTEMP;
                XTEMP:=PTFTHD[I,X];  YTEMP:=PTFTHD[I,Y];
                PTFTHD[I,X]:=(XTEMP-DXB)*C+(YTEMP-DYB)*B;
                PTFTHD[I,Y]:=-(XTEMP-DXB)*B+(YTEMP-DYB)*C;
                PTFTHD[I,Z]:=PTFTHD[I,Z];
            END;
                XTEMP:=PTGC[X];  YTEMP:=PTGC[Y];
                PTGC[X]:=(XTEMP-DXB)*C+(YTEMP-DYB)*B;
                PTGC[Y]:=-(XTEMP-DXB)*B+(YTEMP-DYB)*C;
                XTEMP:=NTGC[X];  YTEMP:=NTGC[Y];
                NTGC[X]:=(XTEMP-DXB)*C+(YTEMP-DYB)*B;
                NTGC[Y]:=-(XTEMP-DXB)*B+(YTEMP-DYB)*C;

            PTIME       := TIME;
            DT  := (PTIME-LTIME)*3600;
            LTIME:= PTIME;
            TTIME:=TTIME+DT;
      END; { OF LOOP FOR FTL MOTION }

   { COMPUTE THE ACTUAL BODY MOVEMENT AND TURNING ANGLE
     BY USING INFORMATION OF TWO SUPPROTING LEGS }

        NEWFT1[X]:=FTHD[3,X];  NEWFT1[Y]:=FTHD[3,Y];
        NEWFT1[Z]:=FTHD[3,Z];  NEWFT2[X]:=FTHD[4,X];
        NEWFT2[Y]:=FTHD[4,Y];  NEWFT2[Z]:=FTHD[4,Z];
        B:=OLDFT1[X]-OLDFT2[X];
```

```
          C:=OLDFT1[Y]-OLDFT2[Y];
          A:=(NEWFT1[X]-NEWFT2[X])/SQRT(B*B+C*C);
          AUXANG:=ATAN2(B,C);
          IF A >= 0.9999 THEN ANGCG:=3.1416/2.-AUXANG
          ELSE
             BEGIN
                  B:=ASIN(A)-AUXANG;
                  C:=3.14159-ASIN(A)-AUXANG;
                  IF ABS(B-TURANG) < ABS(C-TURANG)
                  THEN ANGCG:=B
                  ELSE ANGCG:=C;
             END;
          CG[X]:=OLDFT1[X]-NEWFT1[X]*COS(ANGCG)
                 +NEWFT1[Y]*SIN(ANGCG);
          CG[Y]:=OLDFT1[Y]-NEWFT1[X]*SIN(ANGCG)
                 -NEWFT1[Y]*COS(ANGCG);
          CG[Z]:=OLDFT1[Z]-NEWFT1[Z];
          OLDFT1:=NEWFT1;  OLDFT2:=NEWFT2;

    { COMPUTE THE TOTAL ANGLE AND TOTAL DISTANCE
      THAT BODY CENTER HAS MOVED SO FAR.

          C:=COS(TANGCG);  B:=SIN(TANGCG);
          CGMOVE[X]:=CGMOVE[X]+CG[X]*C-CG[Y]*B;
          CGMOVE[Y]:=CGMOVE[Y]+CG[X]*B+CG[Y]*C;
          CGMOVE[Z]:=CGMOVE[Z]+CG[Z];
          TANGCG:=TANGCG+ANGCG;
          C:=COS(TANGCG);  B:=SIN(TANGCG);
          RESANG:=ANG-ANGCG;
          ANGCG:=ANGCG*180./3.14159;

   { ASSIGN FOOTHOLDS FOR MIDDLE AND REAR LEGS FOR
     THE NEXT LOCOMOTION CYCLE }

30:       IF LEGLIFT=6 THEN
             BEGIN
               FOR I:= -6 TO -3 DO
                  BEGIN
                    J:=-I;
                    FTHD[J,X]:=PTFTHD[J-2,X];
                    FTHD[J,Y]:=PTFTHD[J-2,Y];
                    FTHD[J,Z]:=PTFTHD[J-2,Z];
                  END;
             END
          ELSE;
LTGC[X]:=PTGC[X];  LTGC[Y]:=PTGC[Y];
PTGC[X]:=NTGC[X];  PTGC[Y]:=NTGC[Y];

{ FETCH THE NEXT FOOTHOLD FOR A FRONT FOOT }

SAFE:=FALSE;
WHILE NOT(SAFE) DO
  BEGIN
          NEWDATA;
          FOOTHEIGHT;
          TURNANGLE;
          SYNTHMOTION;
          SAFECK;
          IF STOP = 1 THEN SAFE:=TRUE
          ELSE;
    END;
LTIME:=TIME;
IF STOP <> 1 THEN GOTO 20
ELSE;
NORMALIZE;
END.
```

```
(*$E+*)
        { FILE:  FOOT37.PAS    }

                    /*****************************/
                    /*   PROCEDURE:   FOOTPATH   */
                    /*****************************/

/****************************************************************/
/* PROGRAMMERS: DENNIS PUGH, TED CHANG                          */
/* DATE:        30-MAR-82                                       */
/* MODIFIED BY SHENG-JEN TSAI TO SUIT THE FTL LOCOMOTION.       */
/*                                                              */
/* FUNCTION:    CALCULATES FOOT TRAJECTORIES TO IMPLEMENT       */
/*              THE BODY RATES COMMANDED BY BODY MOTION         */
/*              PLANNING.                                       */
/*                                                              */
/* USER GUIDE:  THE CALLING FORMAT IS:      FOOTPATH;           */
/*                                                              */
/*                                                              */
/* PROCEDURES CALLED:   JSERVO, RBADC                           */
/*                                                              */
/* GLOBAL VARIABLES                                             */
/*      REFERENCED:     VELX,    VELY,   DPSI                   */
/*                      MIDSTX, MIDSTY, MIDSTZ                  */
/*                      DT,      PERIOD, SPERIOD                */
/*                      PHASE,  RPHASE, BETA                    */
/*                      FOOTLIFT                                */
/*                                                              */
/*      MODIFIED:       XPD,     YPD,     ZPD                   */
/*                      XRD,     YRD,     ZRD                   */
/*                      XFD,     YFD,     ZFD                   */
/*                      SUPPORT,   ZPTERM                       */
/*                      DXB,     DYB,     PSIC                  */
/****************************************************************/

PROCEDURE FOOTPATH;

CONST   ATTSCALE = -0.7854;      /* ATTITUDE SCALE FACTOR          */
        ATTPOLE = 1.;            /* POLE FOR ATTITUDE CONTROL      */
        FTOTAL = 310.0;          /* TOTAL VEHICLE WEIGHT ( LBS.)   */
        PWRSCALE = -33333.3;     /* POWER SCALE FACTOR             */

VAR
        ALPHA,          /* CRAB ANGLE WRT LONGITUDINAL AXIS     */
        A,B,C,          /* TEMPARORY VARIABLES                  */
        BH,             /* HEIGHT OF THE BODY                   */
        CGX, CGY,       /* COORDINATES OF CG. PROJECTION        */
        DT1, DT2,       /* BACKED UP TIME FROM MIDSTANCE        */
        DXB1, DYB1,     /* TOUCHDOWN DISPL. FROM MIDSTANCE      */
        DXE, DYE,       /* BODY DISPLACEMENT IN EARTH COORD.    */
        LPHASE,         /* LEG PHASE VARIABLE                   */
        PITCH, ROLL,    /* ACTUAL ANGLES FROM GYRO              */
        PPITCH, PROLL,  /* ACTUAL ANGLES FROM PENDULUM          */
        POWER,          /* HEXAPOD INSTANTANEOUS POWER CONSUMP. */
        PSIC1,          /* BODY ANGULAR DISPLACEMENT OVER DT    */
        Q, R, S,          /* INTERMED. FORCE OPTIMIZATION TERMS   */
        STIME,            /* TIME AT WHICH DATA FRAME STORED      */
        SUMX,   SUMY,     /* SUM OF FOOT DISPLACEMENTS            */
        SUMX2,  SUMY2,    /* SUM OF SQUARES OF FOOT DISPLACEMENTS */
        SUMXY,            /* SUM OF FOOT DISPL. CROSS-PRODUCTS    */
        TIMEFP,           /* REMAINING TIME IN TRANSFER PHASE     */
        TRTIME,           /* TOTAL TIME IN TRANSFER PHASE         */
        VEL,              /* MAGNITUDE OF VELOCITY VECTOR         */
        ZOFF,             /* Z POSITION ERROR FOR ATT. CONTROL    */
        ZRTERM            /* Z RATE OFFSET FOR ATTITUDE CONTROL   */

                : REAL;

        FRSTCH,           /* FIRST A/D CHANNEL FOR DATA FETCH     */
        I,                /* LOOP COUNTER                         */
        N                 /* SUPPORT PHASE LEG SET COUNTER        */

                : INTEGER;

        TOPBLOCK          /* TOP A/D CHANNELS                     */

                : ARRAY18;

PROCEDURE JSERVO( MOVE: BOOLEAN );  EXTERNAL;

PROCEDURE RBADC( NOCHAN, FRSTCH: INTEGER; SCALE: REAL; VAR INDATA: ARRAY18 )
        EXTERNAL;

FUNCTION  DELTATIME: REAL; EXTERNAL;

FUNCTION  ATAN2(Y, X : REAL): REAL;  EXTERNAL;


BEGIN   /* FOOT37 */

/****************************************************************/
/*  EXPRESS INCREMENTAL BODY DISPLACEMENT IN BODY COORDINATES  */
/****************************************************************/

/*** CALCULATE INCREMENTAL BODY DISPLACEMENT WRT GROUND ***/
PSIC    := DPSI * DT;

/*** CALCULATE MAGNITUDE OF VELOCITY WRT GROUND ***/
VEL := VELX * VELX + VELY * VELY;
IF VEL <> 0.0 THEN VEL := SQRT(VEL);

IF ABS(DPSI) > 0.00001
    THEN
        BEGIN
        DXE     := VEL/DPSI * SIN(PSIC);
        DYE     := VEL/DPSI * (1.0 - COS(PSIC));
        END
    ELSE
        BEGIN
        DXE     := VEL * DT;
        DYE     := 0.0;
        END;

/*** ROTATE DISPLACEMENT VECTORS TO BODY COORDINATES ***/
ALPHA := ATAN2( VELY, VELX );
```

```
DXB :=
         DXE * COS(ALPHA)
       - DYE * SIN(ALPHA);

DYB :=   DXE * SIN(ALPHA)
       + DYE * COS(ALPHA);

RBADC(5,58,1,TOPBLOCK);
PITCH:=TOPBLOCK[7]*ATTSCALE;
ROLL:=TOPBLOCK[8]*ATTSCALE;

{ ESTIMATE THE HEIGHT OF THE BODY }

IF ACTION = BODYMOTION THEN
   BEGIN
        BH:=0.;
        FOR I:=1 TO 6 DO
        BEGIN
           BH:=BH+ZPA[I];
        END;
        BH:=BH/6.;
    END
ELSE;

/********************************/
/*  GENERATE FOOT TRAJECTORIES  */
/********************************/
FOR I := 1 TO 6 DO { GENERATE FOOT COORDINATE FOR LEG I }
    BEGIN

    IF NOT( I IN SUPPORT )
        THEN { LEG IN TRANSFER PHASE }
            BEGIN
                A:=2.*(FTHD[I,Z]-PTFTHD[I,Z])
                   -4.*(FOOTLIFT-PTFTHD[I,Z]);
                B:=FTHD[I,Z]-PTFTHD[I,Z]-A;
                C:=PTFTHD[I,Z];
                IF TPHASE > 1. THEN
                  BEGIN
                    XRD[I]:=0.;
                    YRD[I]:=0.;
                    ZRD[I]:=1.;
                    XPD[I]:=XPA[I];
                    YPD[I]:=YPA[I];
                    ZPD[I]:=ZPA[I]+0.5;
                  END
                ELSE
                BEGIN
                IF { LEG AT TOP OF TRANSFER PHASE AND FORCE NOT YET ZEROED }
                (TPHASE > 0.5) AND (FZERO[I] = FALSE)

                THEN  { UPDATE OFFSET FORCES FOR LEG I }
                    BEGIN
                    FRSTCH := ( I - 1 ) * 3;
                    RBADC( 3, FRSTCH, FSCALE, ZEROFORCE);
                    FZERO[I] := TRUE;  { FLAG THAT LEG I IS UPDATED }
                    ZPTERM[I] := 0.0;  { INITIALIZE ATTITUDE CORRECTION TERM }
                    END;

                { END IF }
    /*** CALCULATE TIME LEFT TILL TOUCHDOWN OF THE LEG ***/
        TIMEFP:= (1.-TPHASE)*LIFTTIME;

    /*** COMPUTE DESIRED FOOT POSITION ***/
    IF DT < TIMEFP
        THEN
            BEGIN
            XPD[I] := XPD[I] + (FTHD[I,X] - XPD[I]) * DT/TIMEFP;
            YPD[I] := YPD[I] + (FTHD[I,Y] - YPD[I]) * DT/TIMEFP;
            ZPD[I] := A*TPHASE*TPHASE+B*TPHASE+C;
            END
        ELSE
            BEGIN
            XPD[I] := FTHD[I,X];
            YPD[I] := FTHD[I,Y];
            ZPD[I] := FTHD[I,Z];
            END;

        { END IF DT }

    /*** COMPUTE DESIRED FOOT RATE ***/
    XRD[I] := ( FTHD[I,X] - XPD[I] ) / TIMEFP;
    YRD[I] := ( FTHD[I,Y] - YPD[I] ) / TIMEFP;
    ZRD[I] :=(2.*A*TPHASE+B)/LIFTTIME;
     END; { ELSE TPHASE > 1. }
    END  { TRANSFER PHASE }

ELSE  { FOOT IN SUPPORT PHASE }
    BEGIN
    FZERO[I] := FALSE;  { FLAG THAT FORCE NOT ZEROED THIS CYCLE }
    RSUPPORT[I]:=10.;
    /*** COMPUTE ATTITUDE CONTROL VARIABLES ***/
    ZOFF := -PITCH  * XPA[I] + ROLL * YPA[I];
    IF (ATTITUDE = TRUE)
        THEN
            BEGIN
            ZRTERM:=ATTPOLE*ZOFF;
            ZPTERM[I]:=ZPTERM[I]+ZRTERM*DT;
            IF ABS(ZPTERM[I]) > ABS(ZOFF)
               THEN ZPTERM[I] := ZOFF
               ELSE;
            END
        ELSE
            BEGIN
            ZPTERM[I] := 0.0;
            ZRTERM := 0.0;
            END;

        { END IF ATTITUDE }
    /*** COMPUTE DESIRED FOOT POSITION ***/
    XPD[I] :=    PTFTHD[I,X];

    YPD[I] :=    PTFTHD[I,Y];

  { ADJUST THE HEIGHT OF THE BODY IF NECESSARY }

    IF ACTION = BODYMOTION THEN
      BEGIN
        IF BH < 15. THEN
```

```
                 ZPD[I]:= ZPD[I]+0.5/11.*DT
             ELSE IF BH > 20 THEN
                   ZPD[I]:=ZPD[I]-0.5/11.*DT
                   ELSE ZPD[I]:=ZPD[I];
           END
         ELSE
         ZPD[I] :=    PTFTHD[I,Z]+ ZPTERM[I];

         /*** COMPUTE DESIRED FOOT RATE ***/
         XRD[I] := -VELX + DPSI * YPD[I];
         YRD[I] := -VELY - DPSI * XPD[I];
         ZRD[I] := ZRTERM;

         END; /* SUPPORT PHASE */

     END;  /* FOR I */

/**************************************/
/*  COMPUTE OPTIMAL FORCE SETPOINTS  */
/**************************************/

N := 0;
SUMX := 0.0;
SUMY := 0.0;
SUMX2 := 0.0;
SUMY2 := 0.0;
SUMXY := 0.0;


FOR I := 1 TO 6 DO
     IF I IN SUPPORT THEN
          BEGIN
          SUMX := SUMX + XPA[I];
          SUMY := SUMY + YPA[I];
          SUMX2 := SUMX2 + XPA[I] * XPA[I];
          SUMY2 := SUMY2 + YPA[I] * YPA[I];
          SUMXY := SUMXY + XPA[I] * YPA[I];
          N := N + 1;
          END;  { IF I IN SUPPORT }

    { END FOR I }

R := (SUMX * SUMY2 - SUMXY * SUMY) / (SUMX2 * SUMY2 - SUMXY * SUMXY);
Q := (SUMY - SUMXY * R) / SUMY2;
S := N - ((SUMX * SUMY2 - SUMXY * SUMY) * R + SUMY * SUMY) / SUMY2;

CGX := -SIN( PITCH ) * MIDSTZ;
CGY := SIN( ROLL ) * MIDSTZ;

/*** COMPUTE FOOT FORCE SETPOINTS ***/

FOR I := 1 TO 6 DO
    BEGIN
    XFD[I] := 0.0;
    YFD[I] := 0.0;

    IF I IN SUPPORT
         THEN
```

```
                IF OPTIMIZATION = TRUE
                    THEN ZFD[I] := ( 1 - Q * ( YPA[I] - CGY )
                                       - R * ( XPA[I] - CGX )   ) * FTOTAL / S
                    ELSE ZFD[I] := FTOTAL / N

            ELSE
                ZFD[I] := 0.0;

        END; { FOR I }
        JSERVO( MOVE );                           { CALL SERVO ROUTINE }

    END;  /* FOOTPATH */
```

CHECK.PAS

```
(*$E+*)
        {  FILE CHECK.PAS  }

/****************************************************/
/* PROGRAMMER : SHENG-JEN TSAI                      */
/* DATE : APRIL 20, 1983                            */
/* FUNCTION : GIVEN A FOOTHOLD (XG,YG,ZG) FOR A LEG, */
/*            CHECK IF THE LEG CAN REACH IT WITHOUT */
/*            EXCEEDING THE KINEMATIC MARGINS.      */
/*                                                  */
/* FUNCTIONS CALLED : ATAN2, ASIN, SIGN             */
/*                                                  */
/* USER GUIDE :  >PAS CHECK=CHECK                   */
/*               >MAC CHECK=CHECK                   */
/*                                                  */
/****************************************************/

PROCEDURE  CHECK(XG,YG,ZG :REAL; VAR SAI,TH1,TH2:REAL);

CONST L1=12.56;  L2=17.00; L3=0.;
      L4= 2.5;   L5=2.463;
      XHIP=22.75;  YHIP=8.562;

VAR   S,X,Y,Z,A,B,C,D,E:REAL;

FUNCTION ATAN2(Y,X : REAL) : REAL; EXTERNAL;

FUNCTION ASIN(VALUE : REAL) : REAL; EXTERNAL;

FUNCTION SIGN(X : REAL) :REAL; EXTERNAL;

BEGIN
        Z:=ZG;
        Y:=YG;  X:=XG;
        SAI:=ATAN2(X,Y);
        S:=SIN(SAI);  C:=COS(SAI);
        SAI:=SAI*180./3.14159;
        IF S=0. THEN A:=0.
        ELSE A:=(X-L4*S+L3*C)/S;
        A:=A*A;
        B:=A+Z*Z  1*L1-L2*L2-L5*L5;
        D:=L1*L: _5*L5;
        D:=SQRT(D);
        D:=2.*L2*D;
        B:=B/D;
        IF ABS(B) > 1. THEN B:=SIGN(B)
        ELSE;
        B:=ASIN(B);
        E:=ATAN2(L5,L1);
        TH2:=B-E;
        S:=SIN(TH2);  C:=COS(TH2);
        TH2:=TH2*180./3.14159;
        A:=L5+L2*C;  B:=L1+L2*S;
        C:=A*A+B*B;
        C:=SQRT(C);
        C:=Z/C;
        IF ABS(C) > 1. THEN C:=SIGN(C)
```

```
               ELSE;
               C:=-ASIN(C);
               D:=ATAN2(A,B);
               TH1:=C+D;
               TH1:=TH1*180./3.14159;
          END;
```

CENTER.PAS

```
        { FILE: CENTER.PAS }

/*************************************************************/
/* PROGRAMMER: SHENG-JEN TSAI                               */
/* DATE: APRIL 20, 1983                                     */
/* FUNCTION: GIVEN (X1,Y1,Z1,IX1,IY1) AND (X2,Y2,Z2,IX2,IY2),*/
/*           COMPUTE THE LOCATION OF THE LENS CENTER WITH   */
/*           RESPECT TO THE IMAGE PLANE.                    */
/*                                                          */
/* PROCEDURE CALLED: NONE                                   */
/*                                                          */
/* USER GUIDE:   >PAS CENTER=CENTER                         */
/*               >MAC CENTER=CENTER                         */
/*               >TKB/FP/CP=CENTER,[1,1]PASLIB/LB           */
/*               >RUN CENTER                                */
/*                                                          */
/*************************************************************/

PROGRAM CENTER;

LABEL   10;

TYPE
        MATRIX=ARRAY[1..4,1..4] OF REAL;
        VECTOR=ARRAY[1..3] OF REAL;
CONST
        F=6.5E-3;
        DX=4.6E-5;
        DY=3.6E-5;
        NX=248;
        NY=244;
        A3=-7.073564E2;  A2=6.84817E1;
        A1=-3.2127E-1;  A0=6.1816E-4;
        B3=-7.86339E2;  B2=4.78627E1;
        B1=-1.18877E-1;  B0=-5.3976E-5;
VAR
        TMATRIX:MATRIX;
        C21F,C22F,V21,V22,V1,U1,U2,D:VECTOR;
        I,X1OFF,Y1OFF,X2OFF,Y2OFF:INTEGER;
        X,Y,Z,AC,BC,A,B,C,IR,R,XIM2,YIM2,ZIM2,
        K1,K2,K3,LX,LY,ANG,XIM1,YIM1,ZIM1,
        XC,YC,X1,Y1,Z1,X2,Y2,Z2,XP1,YP1,XP2,YP2:REAL;
```

```
BEGIN

10:     WRITELN('INPUT THE IMAGE COORDINATES XP1,YP1,XP2,YP2');
        READLN(XP1,YP1,XP2,YP2);
        WRITELN('INPUT X1,Y1,Z1');
        READLN(X1,Y1,Z1);
        WRITELN('INPUT X2,Y2,Z2');
        READLN(X2,Y2,Z2);
        D[1]:=X2-X1;  D[2]:=Y2-Y1;  D[3]:=Z2-Z1;
        LX:=NX*DX;   LY:=NY*DY;
        XIM1:=(XP1-0.5)*DX;
        YIM1:=(YP1-0.5)*DY;
        ZIM1:=0.;
        V1[1]:=XIM1-X1;
        V1[2]:=YIM1-Y1;
        V1[3]:=ZIM1-Z1;
        XIM2:=(XP2-0.5)*DX;
        YIM2:=(YP2-0.5)*DY;
        ZIM2:=0.;
        V22[1]:=XIM2-X2;
        V22[2]:=YIM2-Y2;
        V22[3]:=ZIM2-Z2;
        B:=V1[1]*V1[1]+V1[2]*V1[2]+V1[3]*V1[3];
        B:=SQRT(B);
        C:=V22[1]*V22[1]+V22[2]*V22[2]+V22[3]*V22[3];
        C:=SQRT(C);
        FOR I:=1 TO 3 DO
          BEGIN
            U1[I]:=V1[I]/B;
            U2[I]:=V22[I]/C;
          END;
        K1:=U1[1]*D[1]+U1[2]*D[2]+U1[3]*D[3];
        K2:=U1[1]*U2[1]+U1[2]*U2[2]+U1[3]*U2[3];
        K3:=U2[1]*D[1]+U2[2]*D[2]+U2[3]*D[3];
        AC:=(K1-K2*K3)/(1.-K2*K2);
        BC:=(K2*K1-K3)/(1.-K2*K2);
        X:=AC*U1[1]+X1;  Y:=AC*U1[2]+Y1;  Z:=AC*U1[3]+Z1;
        WRITELN('X=',X:10,'  Y=',Y:10,'  Z= ,Z:10);
        XC:=X/DX;  YC:=Y/DY;
        WRITELN('XCENTER=',XC:8:3,'  YCENTER=',YC:8:3);
        X:=BC*U2[1]+X2;
        Y:=BC*U2[2]+Y2;
        Z:=BC*U2[3]+Z2;
        WRITELN('X=',X:10,'  Y=',Y:10,'  Z=',Z:10);
        XC:=X/DX;  YC:=Y/DY;
        WRITELN('XCENTER=',XC:8:3,'  YCENTER=',YC:8:3);
        WRITELN;  WRITELN;
        GOTO 10;
END.
```

ROOT4.PAS

```
(*$E+*)

/*********************************************************/
/* FILE: ROOT4.PAS                                       */
/* PROGRAMMER:  SHENG-JEN TSAI                           */
/* DATE:  MARCH 12, 1983                                 */
/*                                                       */
/* FUNCTION: TO COMPUTE THE POSITIVE, REAL ROOTS OF A 4TH */
/*           ORDER POLYMIAL BY NEWTON-RAPHSON METHOD.  THE */
/*           DESIRED ROOT R MUST BE GREATER THAN THE      */
/*           INITIAL VALUE IR.                            */
/*                                                       */
/* PROCEDURES CALLED: NONE                               */
/*                                                       */
/*********************************************************/

procedure ROOT(k4,k3,k2,k1,k0:real;
               VAR  R,IR:REAL);
VAR
   POFR,PDOFR,KEY,RNEXT:REAL;

BEGIN
        KEY:=1.;
        R:=IR;
        POFR:=1.;
        WHILE ABS(POFR) >= 1.0E-6 DO
        BEGIN
        POFR:=-K4*R*R*R*R-K3*R*R*R-K2*R*R-(K1-1)*R-(K0+IR);
        PDOFR:=-4.*K4*R*R*R-3.*K3*R*R-2.*K2*R-(K1-1);
        KEY:=POFR/PDOFR;
        R:=R-KEY;
        END;
END;
```

FOVF.FTN

```
C       CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C       C PROGRAMMER:   SHENJ-JEN TSAI            C
C       C DATE:         MARCH 12, 1983            C
C       C FUNCTION  COMPUT THE FIELD OF VIEW OF   C
C       C           BOTH CAMERAS AS A FUNCTION OF C
C       C           TILT AND ROLL ANGLES AND PLOT C
C       C           THE RESULT IN THE HP PLOTTER. C
C       C                                         C
C       C SUBROUTINES CALLED: INITZ, HPPLOT, STOVER C
C       C                                         C
C       C USER GUIDE: FOVF/FP/CP=FOVF,HPPLOT,     C
C       C             [1,1]F4POTS/LB              C
C       C             /                           C
C       C             COMMON=IOPAGE:WR            C
C       C             //                          C
C       C                                         C
C       CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

        DIMENSION CENTER(3),UR(3),UL(3),URB(3),ULB(3)
        DIMENSION T(4,4),O(3)
        DIMENSION  IM(2,30),ICA1(2,5),ICA2(2,5)
        REAL   K,LX,LY,LZ,LR(3),LL(3),LRB(3),LLB(3)

C       GENERATE DATA ARRAY IM FOR THE HEXAPOD BODY
C       TO BE PLOTTED IN HP PLOTTER

        IM(1,1)=350
        IM(2,1)=332
        IM(1,2)=425
        IM(2,2)=332
        IM(1,3)=425
        IM(2,3)=216
        IM(1,4)=350
        IM(2,4)=216
        IM(1,5)=425
        IM(2,5)=216
        IM(1,6)=425
        IM(2,6)=100
        IM(1,7)=350
        IM(2,7)=100
        IM(1,8)=IM(1,6)
        IM(2,8)=IM(2,6)
        IM(1,9)=575
        IM(2,9)=100
        IM(1,10)=650
        IM(2,10)=100
        IM(1,11)=IM(1,9)
        IM(2,11)=IM(2,9)
        IM(1,12)=575
        IM(2,12)=216
        IM(1,13)=650
        IM(2,13)=216
        IM(1,14)=IM(1,12)
        IM(2,14)=IM(2,12)
        IM(1,15)=575
        IM(2,15)=332
        IM(1,16)=650
```

```
              IM(2,16)=332
              IM(1,17)=IM(1,2)
              IM(2,17)=IM(2,2)
              N=0
              DO 11 I=1,2
              DO 11 J=1,5
              ICA1(I,J)=0.
              ICA2(I,J)=0.
11            CONTINUE

C             CLEAR THE HP SCREEN AND PLOT THE HEXAPOD BODY

              CALL INITZ
              CALL HPPLOT(ICA1,5,2)
              CALL HPPLOT(ICA2,5,3)
              CALL INITZ
              CALL HPPLOT(IM,17,1)

    1         WRITE(5,5)
    5         FORMAT('INPUT PITCH AND ROLL ANGLES')
              READ(5,*) ANG1,ANG2
              ANG2=ANG2/180.*3.1416
              ANG1=ANG1/180.*3.1416
              N=N+1
              NT=0
              F=6.5E-3
              PX=4.6E-5
              PY=3.6E-5
              DX=-0.13
              DY=0.15
              DZ=-0.525
              LX=0.05
              LY=0.
              LZ=0.

C             COMPUTE THE FIELD OF VIEW OF CAMERA 1
C             AND CAMERA 2

              CENTER(1)=118*PX
              CENTER(2)=119*PY
              CENTER(3)=-F
4             UL(1)=0.
              UL(2)=0.
              UL(3)=0.
              UR(1)=248*PX
              UR(2)=0.
              UR(3)=0.
              LL(1)=0.
              LL(2)=244*PY
              LL(3)=0.
              LR(1)=248*PX
              LR(2)=244*PY
              LR(3)=0.

              NT=NT+1
              DO 10 I=1,3
              UL(I)=UL(I)-CENTER(I)
              UR(I)=UR(I)-CENTER(I)
              LL(I)=LL(I)-CENTER(I)
              LR(I)=LR(I)-CENTER(I)
```

```
10    CONTINUE
         T(1,1)=0.
         T(1,2)=-COS(ANG1)
         T(1,3)=SIN(ANG1)
         T(2,1)=COS(ANG2)
         T(2,2)=-SIN(ANG1)*SIN(ANG2)
         T(2,3)=-COS(ANG1)*SIN(ANG2)
         T(3,1)=SIN(ANG2)
         T(3,2)=SIN(ANG1)*COS(ANG2)
         T(3,3)=COS(ANG1)*COS(ANG2)
      DO 40 I=1,3
      ULB(I)=T(I,1)*UL(1)+T(I,2)*UL(2)+T(I,3)*UL(3)
      URB(I)=T(I,1)*UR(1)+T(I,2)*UR(2)+T(I,3)*UR(3)
      LLB(I)=T(I,1)*LL(1)+T(I,2)*LL(2)+T(I,3)*LL(3)
      LRB(I)=T(I,1)*LR(1)+T(I,2)*LR(2)+T(I,3)*LR(3)
40    CONTINUE

      D(1)=LX*COS(ANG1)+LZ*SIN(ANG1)+DX
      D(2)=LX*SIN(ANG2)*SIN(ANG1)+LY*COS(ANG2)
     1 -LZ*SIN(ANG2)*COS(ANG1)+DY
      D(3)=-LX*SIN(ANG1)*COS(ANG2)+LY*SIN(ANG2)
     1 +LZ*COS(ANG2)*COS(ANG1)+DZ
      K=(0.7-D(3))/ULB(3)
      ULB(1)=D(1)+ULB(1)*K
      ULB(2)=D(2)+ULB(2)*K
      ULB(3)=D(3)+ULB(3)*K
      K=(0.7-D(3))/URB(3)
      URB(1)=D(1)+URB(1)*K
      URB(2)=D(2)+URB(2)*K
      URB(3)=D(3)+URB(3)*K
      K=(0.7-D(3))/LLB(3)
      LLB(1)=D(1)+LLB(1)*K
      LLB(2)=D(2)+LLB(2)*K
      LLB(3)=D(3)+LLB(3)*K
      K=(0.7-D(3))/LRB(3)
      LRB(1)=D(1)+LRB(1)*K
      LRB(2)=D(2)+LRB(2)*K
      LRB(3)=D(3)+LRB(3)*K
      PITCH=ANG1*180./3.1416
      ROLL=ANG2*180./3.1416
      WRITE(5,50) PITCH,ROLL
50    FORMAT(//,'ANG1=',F8.4,'  ANG2=',F8.4)
      WRITE(5,60)LRB(1),LRB(2),LRB(3)
60    FORMAT(/,'LRX=',F8.4,'  LRY=',F8.4,'  LRZ=',F8.4)
      WRITE(5,70) LLB(1),LLB(2),LLB(3)
70    FORMAT(/,'LLX=',F8.4,'  LLY=',F8.4,'  LLZ=',F8.4)
      WRITE(5,80) URB(1),URB(2),URB(3)
80    FORMAT(/,'URX=',F8.4,'  URY=',F8.4,'  URZ=',F8.4)
      WRITE(5,90) ULB(1),ULB(2),ULB(3)
90    FORMAT(/,'ULX=',F8.4,'  ULY=',F8.4,'  ULZ=',F8.4)
      FAC=200.
      IF (NT .EQ. 2) GO TO 7
      ICA1(1,1)=ULB(2)*FAC+500
      ICA1(2,1)=ULB(1)*FAC+332
      ICA1(1,2)=URB(2)*FAC+500
      ICA1(2,2)=URB(1)*FAC+332
      ICA1(1,3)=LRB(2)*FAC+500
      ICA1(2,3)=LRB(1)*FAC+332
      ICA1(1,4)=LLB(2)*FAC+500
      ICA1(2,4)=LLB(1)*FAC+332
      ICA1(1,5)=ICA1(1,1)
      ICA1(2,5)=ICA1(2,1)
C     PLOT THE FIELD OF VIEW OF CAMERA 1
      CALL STOVER(2)
      CALL HPPLOT(ICA1,5,2)
      GO TO 9
7     ICA2(1,1)=ULB(2)*FAC+500
      ICA2(2,1)=ULB(1)*FAC+332
      ICA2(1,2)=URB(2)*FAC+500
      ICA2(2,2)=URB(1)*FAC+332
      ICA2(1,3)=LRB(2)*FAC+500
      ICA2(2,3)=LRB(1)*FAC+332
      ICA2(1,4)=LLB(2)*FAC+500
      ICA2(2,4)=LLB(1)*FAC+332
      ICA2(1,5)=ICA2(1,1)
      ICA2(2,5)=ICA2(2,1)
C     PLOT THE FIELD OF VIEW OF CAMERA 2
      CALL STOVER(3)
      CALL HPPLOT(ICA2,5,3)
9     IF (NT .EQ. 2) GO TO 3
      ANG2=-ANG2
      DY=-0.2
      CENTER(1)=113*PX
      CENTER(2)=112*PY
      GO TO 4
3     IF (N .LE. 20) GO TO 1
      STOP
      END
```

POLYFT.PAS

```
/****************************************************************/
/*    FILE: POLYFT.PAS                                          */
/****************************************************************/

PROGRAM POLYFT;

/****************************************************************/
/* PROGRAMMER: S. J. TSAI                                       */
/* DATE:        JULY 6  1982                                    */
/* FUNCTION:                                                    */
/*      THE FUNCTION OF THIS PROGRAM IS TO FIT A SET OF POINTS BY */
/* USING A POLYNOMIAL. THE NUMBER OF POINTS TO BE FITTED IS 50   */
/* OR LESS,AND THE DEGREE OF THE FITTING POLYNOMIAL IS LIMITED TO*/
/* 19 OR LOWER. HOWEVER,THE RISTRICTION ON THE NUMBER OF DATA    */
/* POINTS CAN BE EASILY REMOVED BY CHANGING THE DIMENSION OF     */
/* ARRAYS USED IN THIS PROGRAM.                                  */
/*                                                              */
/*      THE PARTICULAR ALGORITHM EMPOLYED IN THIS PROGRAM TO FIT */
/* POINTS IS BASED ON "ORTHOGONAL POLYNOMIALS",AND THE ERROR IN  */
/* FITTING IS MINIMUM IN THE LEAST SQUARE SENSE. THE ALGORITHM   */
/* IS OUTLINED AS FOLLOWS.                                       */
/*                                                              */
{     P[N+1](X)=(X-BETA[N])*P[N](X)-GAMA[N]*P[N-1](X)            }
{     BETA[N]=SUM(P[N](X)*P[N](X)*X)/SUM(P[N](X)*P[N](X))         }
{     GAMA[N]=SUM(P[N](X)*P[N](X))/SUM(P[N-1](X)*P[N-1](X))       }
/*     P[0](X)=0.       P[1](X)=1.                               */
/* WHERE P[N](X) IS THE NTH ORTHOGONAL POLYNOMIAL.               */
/*                                                              */
/*      THE DATA POINTS TO BE FITTED ARE STORED IN AN EXTERNAL   */
/* FILE "CLEN.DAT". IT IS THE RESPOSIBILITY OF THE USER TO       */
/* CREAT SUCH A FILE BEFORE RUNNUIG THIS PROGRAM.                */
/*                                                              */
/* USER'S GUIDE:                                                */
/*       >PAS POLYFT=POLYFT                                      */
/*       >MAC POLYFT=POLYFT                                      */
/*       >TKB POLYFT/FP/CP=POLYFT,[1,1]PASLIB/LB                 */
/*       >RUN POLYFT                                             */
/*                                                              */
/****************************************************************/

TYPE
    ARRAY80=ARRAY[1..80]OF REAL;
    INDEX=1..81;
    MATRIX=ARRAY[INDEX,INDEX]OF REAL;
    ITEM=
        RECORD
          X:REAL;
          Y:REAL
        END;
    PATH=FILE OF ITEM;

VAR
    ACOE,BETA,GAMA,X,Y,FA,FB,COE,YCAL:ARRAY80;
    YY,STD,SUM,SUMX,SUMY,SUMZ,PD,FF,DIF,PA,PB,PC:REAL;
    I,J,K:INDEX;
    P:MATRIX;
    N,NP,M1,SIZE:INTEGER;
```

```
     POINT:ITEM;
     CLEN1:PATH;

BEGIN
      /****************************************************/
      /* READ IN THE DATA POINTS TO BE FITTED FROM THE      */
      /* EXTERNAL FILE "CLEN1.DAT"                          */
      /****************************************************/

      SIZE:=1;
      WRITELN('INPUT THE NUMBER OF POINTS TO BE FITTED , M1');
      READLN(M1);
      RESET(CLEN1,'CLEN1.DAT','CLEN1.DAT',SIZE);
      FOR I:=1 TO M1 DO
        BEGIN
         POINT:=CLEN1^;
         X[I]:=POINT.X;
         Y[I]:=POINT.Y;
         GET(CLEN1);
        END;
      WRITELN('INPUT THE DEGREE OF POLYNOMIAL , N');
      READLN(N);
      NP:=N+1;

      /**************************************/
      /* BEGINNING OF THE ALGORITHM.        */
      /**************************************/

      SUMY:=0.;   SUMX:=0.;
      FOR J:=1 TO M1 DO
      BEGIN
         SUMY:=SUMY+Y[J];
         SUMX:=SUMX+X[J];
         FA[J]:=1.;   FB[J]:=0.;
      END;
      COE[1]:=SUMY/M1;
      BETA[1]:=SUMX/M1;
      GAMA[1]:=0.;
      FOR J:=1 TO M1 DO
      BEGIN
         YCAL[J]:=COE[1];
      END;
      PD:=M1;
      IF NP >= 2 THEN
         BEGIN
           FOR I:=2 TO NP DO
            BEGIN
             SUMY:=0.;  SUMX:=0.;   SUMZ:=0.;
             FOR J:=1 TO M1 DO
             BEGIN
                 IF I <= 2 THEN FF:=X[J]-BETA[I-1]
                 ELSE FF:=(X[J]-BETA[I-1])*FA[J]
                          -GAMA[I-1]*FB[J];
                 FB[J]:=FA[J];
                 FA[J]:=FF;
                 PB:=FF*FF;
                 PA:=PB*X[J];
                 PC:=FF*Y[J];
                 SUMY:=SUMY+PA;
                 SUMX:=SUMX+PB;
```

```
                SUMZ: SU Z+PC;
            END;  /* END J  */
        BETA[I]:=SUMY/SUMX;
        GAMA[I]:=SUMX/PD;
        COE[I]:=SUMZ/SUMX;
        PD:=SUMX;
        SUM:=0.;
        FOR J:=1 TO M1 DO
            BEGIN
                YCAL[J]:=YCAL[J]+COE[I]*FA[J];
                DIF:=Y[J]-YCAL[J];
                SUM:=SUM+DIF*DIF;
            END;  /* END J  */
        STD:=SQRT(SUM/M1);
      END;  /* END I  */
  END    /* END  THEN   */
  ELSE
     BEGIN
        YY:=COE[1];
        SUM:=0.;
        FOR I:=1 TO M1 DO
           BEGIN
                DIF:=Y[I]-YY;
                SUM:=SUM+DIF*DIF;
           END;
        STD:=SQRT(SUM/M1);
     END;

/*********************************************************/
/* COMPUTE THE COEFFICIENTS OF THE POLYNOMIAL            */
/*********************************************************/
   FOR K:=1 TO NP DO
     BEGIN
        P[1,K]:=0.;
     END;
   P[1,1]:=1.;
   FOR I:=2 TO NP DO
     BEGIN
        FOR J:=1 TO I DO
          BEGIN
                IF I=2 THEN P[I,J]:=-BETA[I-1]*P[I-1,J]
                ELSE P[I,J]:=-BETA[I-1]*P[I-1,J]
                              -GAMA[I-1]*P[I-2,J];
                IF J >= 2 THEN P[I,J]:=P[I,J]+P[I-1,J-1]
                ELSE;
          END;  /* END  J  */
     END;  /* END  I  */
   FOR J:=1 TO NP DO
     BEGIN
        ACOE[J]:=0.;
        FOR I:=1 TO NP DO
          BEGIN
                ACOE[J]:=ACOE[J]+COE[I]*P[I,J];
          END;  /*  END  I  */
     END;  /*  END  J  */

   WRITELN('THE DEGREE OF POLYNOMIAL IS',N);
   WRITELN('THE STANDARD DEVIATION =',STD:10);
   WRITELN('THE COEFFICIENTS OF THE POLYNOMIAL ARE :');
   FOR I:=1 TO NP DO
  BEGIN
     J:=I-1;
     WRITELN('A',J:2,'=',ACOE[I]:10);
  END;
END.
```

TRION.PAS

```
(*$E+*)

/*****************************************************/
/* FILE: TRION.PAS                                    */
/* PROGRAMMER:  SHENJ-JEN TSAI                        */
/* DATE:  MARCH 12, 1983                              */
/*                                                    */
/* FUNCTION: TO COMPUTE THE 3-DIMENSIONAL COORDINATES */
/*           (XT,YT,ZT) OF AN OBJECT POINT FROM ITS 2-D */
/*           IMAGE COORDINATES (XP1,YP1,XP2,YP2) BY   */
/*           TRIANGULATION.                           */
/*                                                    */
/* PROCUDURES CALLED: ROOT                            */
/* GLOBAL FILE: GBLTRI.PAS                            */
/*                                                    */
/* USER GUIDE: >PAS TRION=GBLTRI,TRION                */
/*                                                    */
/*****************************************************/

PROCEDURE TRION;
TYPE
        VECTOR=ARRAY[1..3] OF REAL;
CONST
        F=6.5E-3;
        DX=4.6E-5;
        DY=3.6E-5;
        NX=248;
        NY=244;
VAR
        C21F,C22F,V21,V22,V1,U1,U2,D:VECTOR;
        I,X1OFF,Y1OFF,X2OFF,Y2OFF:INTEGER;
        AC,BC,A,B,C,IR,R,PCTERR,
        K1,K2,K3,XPCET,YPCET,ZPCET,LX,LY,ANG,
        RX,RY,RZ:REAL;
        A4,A3,A2,A1,A0,B4,B3,B2,B1,B0:REAL;
        X1,Y1,Z1,X2,Y2,Z2:REAL;

PROCEDURE ROOT(K4,K3,K2,K1,K0:REAL;
                VAR  R,IR:REAL); EXTERNAL;

FUNCTION  ATAN2(  Y,X:REAL):REAL; EXTERNAL;

BEGIN
        A4:=-6.420006E5;  A3:=7.877979E3;  A2:=1.697816E1;
        A1:=-1.114581E-1;  A0:=1.17950E-4;
        B4:=-6.432071E5;
        B3:=8.959416E3;  B2:=3.917839; B1:=-6.724151E-2;
        B0:=8.380797E-5;
        A:=(XP1-119)*DX;   B:=(YP1-118)*DY;
        IR:=SQRT(A*A+B*B);
        IF (IR > 2.5E-3) THEN
           BEGIN
                ANG:=ATAN2(B,A);
                ROOT(A4,A3,A2,A1,A0,R,IR);
                A:=R*COS(ANG);  B:=R*SIN(ANG);
           END
        ELSE;
```

```
            V1[1]:=A;
            V1[2]:=B;
            V1[3]:=F;
            A:=(XP2-111)*DX;  B:=(YP2-113)*DY;
            IR:=SQRT(A*A+B*B);
            IF (IR > 2.5E-3) THEN
               BEGIN
                     ANG:=ATAN2(B,A);
                     ROOT(B4,B3,B2,B1,B0,R,IR);
                     A:=R*COS(ANG);  B:=R*SIN(ANG);
               END
            ELSE;
            V22[1]:=A;
            V22[2]:=B;
            V22[3]:=F;
            C22F[1]:=DX*112;
            C22F[2]:=DY*113;
            C22F[3]:=-F;
            FOR I:=1 TO 3 DO
              BEGIN
                V21[I]:=TMATRIX[I,1]*V22[1]+TMATRIX[I,2]*V22[2]
                         +TMATRIX[I,3]*V22[3];
                C21F[I]:=TMATRIX[I,1]*C22F[1]+TMATRIX[I,2]*C22F[2]
                          +TMATRIX[I,3]*C22F[3]+TMATRIX[I,4];
              END;
            B:=V1[1]*V1[1]+V1[2]*V1[2]+V1[3]*V1[3];
            B:=SQRT(B);
            C:=V21[1]*V21[1]+V21[2]*V21[2]+V21[3]*V21[3];
            C:=SQRT(C);
            D[1]:=TMATRIX[1,4];  D[2]:=TMATRIX[2,4];
            D[3]:=TMATRIX[3,4];
            FOR I:=1 TO 3 DO
              BEGIN
                U1[I]:=V1[I]/B;
                U2[I]:=V21[I]/C;
              END;
            K1:=U1[1]*D[1]+U1[2]*D[2]+U1[3]*D[3];
            K2:=U1[1]*U2[1]+U1[2]*U2[2]+U1[3]*U2[3];
            K3:=U2[1]*D[1]+U2[2]*D[2]+U2[3]*D[3];
            AC:=(K1-K2*K3)/(1.-K2*K2);
            BC:=(K2*K1-K3)/(1.-K2*K2);
            X1:=AC*U1[1]+DX*119;  Y1:=AC*U1[2]+DY*118;
            Z1:=AC*U1[3]-F;
            X2:=BC*U2[1]+C21F[1];
            Y2:=BC*U2[2]+C21F[2];
            Z2:=BC*U2[3]+C21F[3];
            XT:=(X1+X2)/2.;  YT:=(Y1+Y2)/2.;  ZT:=(Z1+Z2)/2.;
END;
```

VID.PAS

```
/*******************************************************/
/* FILE: VID.PAS                                       */
/* PROGRAMMER:  SHENG-JEN TSAI                         */
/* DATE:  MARCH 12, 1983                               */
/*                                                     */
/* FUNCTION: TO TEST THE NORMAL OPERATION OF THE VISION*/
/*           SYSTEM.  WHEN THIS PROGRAM IS EXECUTED,   */
/*           THE 11/70 COMPUTER WILL CONTROL THE       */
/*           SCANNING OF THE VISION SYSTEM BY SENDING  */
/*           ONE BYTE OF DATA TO TRIGGER THE IMAGE     */
/*           DETECTION EVENT.  THE RESULTING 5 BYTES   */
/*           OF DATA ARE SENT BACK TO 11/70 FOR        */
/*           TRIANGULATION.  IF NO IMAGE IS SENSED OR  */
/*           THE DETECTED IMAGE IS OUT OF REACH FOR    */
/*           THE HEXAPOD, MESSAGE WILL BE DISPLAYED ON */
/*           THE TERMINAL SCREEN TO PROMPT FOR ANOTHER */
/*           TRY.                                      */
/*                                                     */
/* PROCEDURES CALLED: TRION                            */
/* FUNCTIONS CALLED: ATAN2,SIGN,ASIN                   */
/*                                                     */
/* USER GUIDE: TKB @VID.CMD                            */
/*                                                     */
/*******************************************************/

   LABEL 10,20;
   CONST  CST=377B;
          HEIGHTT='';
          HEIGHTB='';
          RESETSCREEN='';
          PX1T='';
          px1b='';
          py1t='';
          py1b='';
          pz1t='';
          pz1b='';
          px2t='';
          px2b='';
          py2t='';
          py2b='';
          PNOt='';
          pnob='';
          POUTT='';
          POUTB='';
          wide='';
          invvideo='';
          remvideo='';
          BELL='';
          clear='';
          pclear='';

   VAR    READY,IMAGE,N,I,J,A,B,SB,X1,Y1,X2,Y2:INTEGER;
          XE,YE,ZE,ANG,ANG1,ANG2,ANG3:REAL;
          T,T1B,TB1,TB2,T12:MATRIX;
(*$C
          .MCALL ALUN$S,QIOW$S,QIO$S,ASTX$S,WTSE$S,EXIT$S
          .EVEN
STAD:     .BYTE    TC.TBF^1
RDBUF:    .BLKB  6
IOST:     .BLKW  2
WRBUFC:   .BYTE   ^B00000010
WRBUF:    .BYTE   ^B00000001
          .EVEN      *)

PROCEDURE TRION; EXTERNAL;

PROCEDURE CHECK; { TO CHECK IF THE IMAGE IS OUT
                   OF REACH FOR THE HEXAPOD   }

   CONST L1=12.56;  L2=17.00; L3=0.;
         L4= 2.5;   L5=2.463;
         XHIP=22.75;  YHIP=8.562;

   VAR   X,Y,Z,A,B,C,D,E,SAI,TH1,TH2,S:REAL;

   FUNCTION ATAN2(Y,X : REAL) : REAL; EXTERNAL;

   FUNCTION ASIN(VALUE : REAL) : REAL; EXTERNAL;

   FUNCTION SIGN(X : REAL) :REAL; EXTERNAL;

   BEGIN
        X:=XE;  Z:=ZE-10.5;
        IF YE < 0. THEN Y:=ABS(YE)
        ELSE Y:=YE;
        Y:=Y-YHIP;

     { COMPUTE THE ROTATIONAL ANGLE ANG1 }

        SAI:=ATAN2(X,Y);
        S:=SIN(SAI);  C:=COS(SAI);
        SAI:=SAI*180./3.14159;
        ANG1:=ABS(SAI);

     { COMPUTE THE KNEE JOINT ANGLE ANG2 }

        IF S=0. THEN A:=0.
        ELSE A:=(X-L4*S+L3*C)/S;
        A:=A*A;
        B:=A+Z*Z-L1*L1-L2*L2-L5*L5;
        D:=L1*L1+L5*L5;
        D:=SQRT(D);
        D:=2.*L2*D;
        B:=B/D;
        IF ABS(B) > 1. THEN B:=SIGN(B)
        ELSE;
        B:=ASIN(B);
        E:=ATAN2(L5,L1);
        TH2:=B-E;
        S:=SIN(TH2);  C:=COS(TH2);
        TH2:=TH2*180./3.14159;
        ANG2:=ABS(TH2);

     { COMPUTE THE ELEVATION ANGLE ANG3 }

        A:=L5+L2*C;  B:=L1+L2*S;
```

```
                C:=A*A+B*B;
                C:=SQRT(C);
                C:=Z/C;
                IF ABS(C) > 1. THEN C:=SIGN(C)
                ELSE;
                C:=-ASIN(C);
                D:=ATAN2(A,B);
                TH1:=C+D;
                TH1:=TH1*180./3.14159;
                ANG3:=ABS(TH1);
            END; { OF CHECK }

        BEGIN
                TB1[1,1]:=0.0995; TB1[1,2]:=-0.9355; TB1[1,3]:=0.341;
                TB1[1,4]:=-0.011;
                TB1[2,1]:=0.995; TB1[2,2]:=0.0842; TB1[2,3]:=-0.0341;
                TB1[2,4]:=0.14;
                TB1[3,1]:=0.; TB1[3,2]:=0.343; TB1[3,3]:=0.9392;
                TB1[3,4]:=-0.48;
                TB1[4,1]:=0.; TB1[4,2]:=0.; TB1[4,3]:=0.; TB1[4,4]:=1.;
                TB2[1,1]:=-0.0995; TB2[1,2]:=-0.9355; TB2[1,3]:=0.341;
                TB2[1,4]:=-0.011;
                TB2[2,1]:=0.995; TB2[2,2]:=-0.0842; TB2[2,3]:=0.0341;
                TB2[2,4]:=-0.163;
                TB2[3,1]:=0.; TB2[3,2]:=0.343; TB2[3,3]:=0.9392;
                TB2[3,4]:=-0.48;
                TB2[4,1]:=0.; TB2[4,2]:=0.; TB2[4,3]:=0.; TB2[4,4]:=1.;
                FOR I:=1 TO 3 DO
                   BEGIN
                         T1B[1,I]:=TB1[I,1];  T1B[2,I]:=TB1[I,2];
                         T1B[3,I]:=TB1[I,3];
                         T1B[I,4]:=-(TB1[1,4]*TB1[1,I]+TB1[2,4]*TB1[2,I]
                                    +TB1[3,4]*TB1[3,I]);
                   END;
                T1B[4,1]:=0.;  T1B[4,2]:=0.;  T1B[4,3]:=0.;  T1B[4,4]:=1.;
                FOR I:=1 TO 3 DO
                   BEGIN
                     FOR J:= 1 TO 4 DO
                        BEGIN
                          T12[I,J]:=T1B[I,1]*TB2[1,J]+T1B[I,2]*TB2[2,J]
                                   +T1B[I,3]*TB2[3,J]+T1B[I,4]*TB2[4,J];
                        END;  { OF J }
                   END;  { OF I }
                TMATRIX:=T12;

                 WRITELN(RESETSCREEN);
                 WRITELN(PY1T,INVVIDEO,WIDE,HEIGHTB,HEIGHTT,
                ' SPECIFY FOOTHOLD ');
                 WRITELN(PY1B,INVVIDEO,WIDE,HEIGHTT,HEIGHTB,
                ' SPECIFY FOOTHOLD ');
                 WRITELN(BELL);  WRITELN(BELL);  WRITELN(BELL);
                 WRITELN(REGVIDEO);
        10:       A:=0;
        (*$C      EF1=7
                  LUN=^D9
                  ALUN$S #LUN,#"OT,#0
                  QIOW$S #IO.ATT,#LUN
                  QIOW$S #SF.GMC,#LUN,#EF1,,,,<#STAD,#2>

                  MOV   STAD,A(R5)
```

```
          SWAB   A(R5)      *)
   A:=A AND CST;
    IF A > 0 THEN
               BEGIN
                WRITELN(' UNSOLICITED DATA ', A:3,
                ' BYTES EXISTS ! ');
                (*$C   EXIT$S             *)
               END
             ELSE;

    (*$C
          QIOW$S #IO.WAL,#LUN,#EF1,,#IOST,,<#WRBUF,#1>
          QIOW$S  #IO.RAL!TF.RNE,#LUN,#EF1,,#IOST,,<#RDBUF,#5>
          MOVB  RDBUF,SB(R5)
          MOVB  RDBUF+1,X1(R5)
          MOVB  RDBUF+2,Y1(R5)
          MOVB  RDBUF+3,X2(R5)
          MOVB  RDBUF+4,Y2(R5)  *)

XP1:=X1 AND CST;
YP1:=Y1 AND CST;
XP2:=X2 AND CST;
YP2:=Y2 AND CST;
SB:=SB AND CST;
A:=SB AND 8;
IF A = 0 THEN
         GOTO 20
ELSE;
READY:=SB AND 1;
IF READY=0 THEN GOTO 10
       ELSE;
A:=SB AND 4;  B:=SB AND 16;
IF (A=0) OR (B=0) THEN
        BEGIN
        WRITELN(RESETSCREEN);
        WRITELN(PnoT,INVVIDEO,WIDE,HEIGHTB,HEIGHTT,
        '  NO IMAGE, TRY AGAIN !');
        WRITELN(PnoB,INVVIDEO,WIDE,HEIGHTT,HEIGHTB,
        '  NO IMAGE, TRY AGAIN !');
        WRITELN(BELL);
        (*$C
          QIOW$S #IO.WAL,#LUN,#EF1,,#IOST,,<#WRBUFC,#1>
             *)
        GOTO 10;
        END
       ELSE
        BEGIN
         TRION;
         XE:=XT*TB1[1,1]+YT*TB1[1,2]+ZT*TB1[1,3]+TB1[1,4];
         YE:=XT*TB1[2,1]+YT*TB1[2,2]+ZT*TB1[2,3]+TB1[2,4];
         ZE:=XT*TB1[3,1]+YT*TB1[3,2]+ZT*TB1[3,3]+TB1[3,4];
         XE:=XE/0.0254;  YE:=YE/0.0254;  ZE:=ZE/0.0254;

         WRITELN(RESETSCREEN);
         WRITELN(' X1=',XP1,'  Y1=',YP1,'  X2=',XP2,'  Y2=',YP2);
         writeln(px1t,invvideo,wide,heightb,heightt,
         '  X=',XE:8:4,'  INCH  ');
         WRITELN(PX1B,INVVIDEO,WIDE,HEIGHTT,HEIGHTB,
         '  X=',XE:8:4,'  INCH  ');
```

```
          WRITELN(PY1T,INVVIDEO,WIDE,HEIGHTB,HEIGHTT,
          '  Y=',YE:8:4,'  INCH  ');
          WRITELN(PY1B,INVVIDEO,WIDE,HEIGHTT,HEIGHTB,
          '  Y=',YE:8:4,'  INCH  ');
          WRITELN(Pz1T,INVVIDEO,WIDE,HEIGHTB,HEIGHTT,
          '  Z=',ZE:8:4,'  INCH  ');
          WRITELN(Pz1B,INVVIDEO,WIDE,HEIGHTT,HEIGHTB,
          '  Z=',ZE:8:4,'  INCH  ');
          CHECK;
          IF (ANG1 > 75.) OR (ANG2 > 75.) OR (ANG3 > 75.) THEN
             BEGIN
                 WRITELN(BELL);  WRITELN(BELL);
                 WRITELN(POUTT,INVVIDEO,WIDE,HEIGHTB,HEIGHTT,
                         '  OUT OF REACH, TRY AGAIN ! ');
                 WRITELN(POUTB,INVVIDEO,WIDE,HEIGHTT,HEIGHTB,
                         '  OUT OF REACH, TRY AGAIN ! ');
             END
          ELSE;
     (*$C
           GIOW$S #IO.WAL,#LUN,#[illegible]1, #IOST,,<#WRBUFC,#1>
          *)
           GOTO 10;
         END;
20:      WRITELN(REGVIDEO);
         writeln('');
END.
```

CAC1.PAS

```
/*********************************************************/
/* FILE:  CAC1.PAS                                       */
/* PROGRAMMER:  SHENG-JEN TSAI                           */
/* DATE:   MARCH 12, 1983                                */
/*                                                       */
/* FUNCTION: TO READ THE 5-TUPLE (RX,RY,RZ,IX,IY) FROM AN */
/*           EXTERNAL FILE CAC1.DAT, COMPUTE THE DISTORTED*/
/*           IMAGE RADIUS IR, THE UNDISTORTED RADIUS RR,  */
/*           AND THE AMOUNT OF DISTORTION DR.  THE VALUS  */
/*           OF RR AND DR ARE STORED IN A SECOND EXTERNAL */
/*           CLEN1.DAT WHICH IS TO BE USED BY THE CURVE   */
/*           FITTING ROUTINE PLYFT1.PAS.                  */
/*                                                       */
/* PROCEDURES CALLED: ROOT ( IN FILE ROOT4.PAS )         */
/* FUNCTION CALLED: ATAN ( IN FILE ATAN.PAS)             */
/*                                                       */
/* USER GUIDE: CAC1/FP/CP=CAC1,ROOT4,ATAN,[1,1]PASLIB/LB */
/*                                                       */
/*********************************************************/

PROGRAM CAC1;

CONST
        F=6.5E-3;
        DX=4.6E-5;  DY=3.6E-5;
        NX=248;     NY=244;
TYPE
     OBJECT=
       RECORD
         IX:INTEGER;
         IY:INTEGER;
         RX:REAL;
         RY:REAL
       END;
     BLOCK=FILE OF OBJECT;
     ITEM=
       RECORD
         RR:REAL;
         DR:REAL
       END;
     GROUP=FILE OF ITEM;

PROCEDURE ROOT(K4,K3,K2,K1,K0:REAL;
                VAR  R,IR:REAL); EXTERNAL;

FUNCTION ATAN(Y,X:REAL):REAL; EXTERNAL;

VAR
        CR,M,RR,X,Y,A,Z,B,IR,DR,XUD,YUD,C,
        RX,RY,K4,K3,K2,K1,K0,DIF,ANG1,ANG2:REAL;
        SIZE,IX,IY,IXC,IYC,IXUD,IYUD:INTEGER;
        BUFFER:OBJECT;
        CAC1:BLOCK;
        CELL:ITEM;
        CLEN1:GROUP;

BEGIN
```

```
        WRITELN(' INPUT Z DISTANCE ');
        READLN(Z);
        SIZE:=1;
        REWRITE(CLEN1,'CLEN1.DAT','CLEN1.DAT',SIZE);
        RESET(CAC1,'CAC1.DAT','CAC1.DAT',SIZE);
        WRITELN('RX   ','   RY  ','  IX   ',
              '  IY ',' IXUD ',' IYUD ',
              ' IXC ',' IYC ');

/*** READ RX,RY,IX,IY FROM FILE CAC1.DAT   ***/

        WHILE NOT EOF(CAC1) DO
        BEGIN
        BUFFER:=CAC1^;
        IX:=BUFFER.IX;
        IY:=BUFFER.IY;
        RX:=BUFFER.RX;
        RY:=BUFFER.RY;
        RX:=RX*2.54;
        GET(CAC1);

/*** COMPUTE THE DISTORTED IMAGE RADIUS IR  ***/

        A:=(IX-119)*DX;
        B:=(IY-118)*DY;
        C:=A;
        ANG1:=ATAN(B,A);
        IR:=SQRT(A*A+B*B);

/*** COMPUTE THE UNDISTORTED IMAGE RADIUS RR ***/

        K4:=-6.420006E5;  K3:=7.877979E3;
        K2:=1.697816E1;
        K1:=-1.114581E-1;  K0:=1.179560E-4;
        ROOT(K4,K3,K2,K1,K0,CR,IR);
        M:=Z/F;
        X:=(RX*0.01+DX*119*M)/(M+1);
        Y:=(RY*0.01+DY*118*M)/(M+1);
        XUD:=X/DX;
        YUD:=Y/DY;
        IXUD:=ROUND(XUD);
        IYUD:=ROUND(YUD);
        X:=X-DX*119;
        Y:=Y-DY*118;
        ANG2:=ATAN(Y,X);
        ANG2:=ANG2*180./3.1416;
        RR:=SQRT(X*X+Y*Y);
        A:=CR*COS(ANG1)/DX;
        B:=CR*SIN(ANG1)/DY;
        IXC:=118+ROUND(A);
        IYC:=118+ROUND(B);
        IF C < 0. THEN IYC:=IYC+2
        ELSE;
        DR:=RR-IR;
        DR:=ABS(DR);
        DIF:=RR-CR;

/*** STORE RR AND DR INTO FILE CLEN1.DAT   ***/

        CELL.RR:=RR;
```

```
            CELL.DR:=DR;
            CLEN1^:=CELL;
            PUT(CLEN1);
            WRITELN;WRITELN;
            WRITELN(RX:5:2,'   ',RY:5:2,' ',IX:5,' ',
                    IY:5,' ',IXUD:5,' ',IYUD:5,' ',
                    IXC:5,' ',IYC:5);
            END;
     END.
```

IM2OB.PAS

```
PROGRAM IM2OB;
/*******************************************************************/
/* PROGRAMMER: S. J. TSAI                                          */
/* DATE      : MAY 5 1982                                          */
/*                                                                 */
/*     THIS PROGRAM IS THE "INVERSE PROGRAM" OF OB2IM, IT TAKES    */
/* THE IMAGE COORDINATE (IX,IY) AND COMPUTE THE THREE DIMENSIONAL */
/* SPATIAL COORDINATE (X,Y,Z) OF THE OBJECT POINT,ASSUMING THAT    */
/* THE OBJECT POINT LIES ON A PLANE WHICH IS PERPENDICULAR TO THE */
/* OPTICAL AXIS OF THE CAMERA AND THAT THE DISTANCE BETWEEN THE    */
/* LEN CNETER AND THE OBJECT PLANE, Z ,IS KNOWN.                   */
/*                                                                 */
/*******************************************************************/

CONST
   F=6.5E-3;
   DY=3.6E-5;
   DX=4.6E-5;
VAR
   LX,LY,X,Y,Z,M:REAL;
   IX,IY,I:INTEGER;

BEGIN
        LX:=248.*DX;
        LY:=244.*DY;
        Z:=1.;
        WHILE Z <= 2. DO
         BEGIN
           PAGE(OUTPUT);
           WRITELN('        Z    ','  IX  ','     X     ',
                  '  IY  ','    Y     ');
           M:=Z/F;
           IX:=2;
           IY:=2;
           FOR I:=1 TO 120 DO
            BEGIN
                 X:=DX*IX+(DX*IX-LX/2.)*M;
                 Y:=DY*IY+(DY*IY-LY/2.)*M;
                 WRITELN(Z:10:3,IX:5,X:10:3,IY:5,Y:10:3);
                 IX:=IX+2;
              IY:=IY+2;
            END;    /*   END IY   */
           Z:=Z+0.1;
        END;   /* END Z   */
        END.   /* END IM2OB   */
```

OB2IM.PAS

```
PROGRAM OB2IM;
/*****************************************************************/
/* PROGRAMMER: S. J. TSAI                                        */
/* DATE      : MAY 5 1982                                        */
/*                                                               */
/*      THIS PROGRAM RELATES A THREE DIMENSIONAL POINT WHOSE     */
/* COORDINATE IS (X,Y,Z) TO ITS IMAGE ,(IX,IY),ON THE FOCAL PLANE */
/* THIS INFORMATION IS USEFUL TO CAMERA CALIBRATION.GIVEN ANY    */
/* POINT (X,Y,Z) ON A PLANE WHICH IS PERPENDICULAR TO THE OPTICAL */
/* AXIS OF THE CAMERA,THE CORRESPONDING IMAGE COORDINATE (IX,IY)  */
/* IS COMPUTED.NOTE THAT IX DEPENDS ONLY ON X AND Z,WHILE IY      */
/* DEPENDS ONLY ON Y AND Z.                                       */
/*                                                               */
/*      PARAMETERS:                                              */
/* F   : FOCAL LENGTH OF THE LENS USED,                          */
/* DX  : THE WIDTH OF A PIXEL,                                   */
/* DY  : THE HEIGHT OF A PIXEL,                                  */
/* NX  : NUMBER OF PIXELS IN A ROW,                              */
/* NY  : NUMBER OF ROWS IN A FRAME.                              */
/*                                                               */
/*****************************************************************/

CONST
   F=6.5E-3;
   DY=3.6E-5;
   DX=4.6E-5;
   NX=248;
   NY=244;
VAR
   PX,PY,LX,LY,X,Y,Z,M:REAL;
   IX,IY:INTEGER;

BEGIN
        LX:=NX*DX;
        LY:=NY*DY;
        Z:=1.;
        WHILE Z <= 2. DO
          BEGIN
             PAGE(OUTPUT);
        WRITELN('        Z      ','     X      ','  IX   ',
                '     Y     ','  IY   ');
            M:=Z/F;
            Y:=-1.;
            X:=-1.;

             WHILE Y <= 1. DO
                BEGIN
                          PX:=((X+LX*M/2.)/(M+1))/DX;
                          IX:=ROUND(PX);
                          PY:=((Y+LY*M/2.)/(M+1))/DY;
                          IY:=ROUND(PY);
                          WRITELN(Z:10:3,X:10:3,IX:5,Y:10:3,IY:5);
                          X:=X+0.01;
                          Y:=Y+0.01;
                END;  /* END Y    */
           Z:=Z+0.1;
           END;  /* END Z  */
 END.   /* END OB2IM  */
```

END

DATE
FILMED

9 — 83

DTIC